

Technikerarbeit 2022/23

Energiemonitoring an der EST

Verfasser: Jonas Zeug

Technikerarbeit 2022/23

Informationstechnik

Energiemonitoring an der EST

Verfasser: Jonas Zeug

Betreuer: Martin Rösner (Schule)
Hansjörg Rixner (Firma)

Elektronikschule - Oberhofer Straße 25 - 88069 Tettnang

Projektbeschreibung

Nach einer Veranstaltung an der EST, bei der es um die Probleme und Lösungen der aktuellen Energiewende ging, haben wir im kleinen Kreis überlegt, wie auch die EST mit gutem Beispiel vorangehen kann. Wir kamen schnell zu dem Schluss, dass man Energie nur einsparen kann, wenn man überhaupt weiß, wie und wo diese Energie verbraucht wird. Die EST hat seit über 20 Jahren einen sogenannten Energiespiegel, dieser hängt im Eingangsbereich. Dort werden die Verbrauchswerte wie Strom, Wasser, Gas und Solar angezeigt. Leider ist dieser mittlerweile nicht mehr funktionsfähig. Die Anzeigen sind defekt und Werte wie Wasser und Gas fehlen komplett.

So entstand der Grundstein für diese Technikerarbeit: Den alten Energiespiegel durch ein Dashboard in Form einer Webseite zu ersetzen. Dies sollte durch einen Raspberry Pi geschehen, der die Daten von der alten Steuerung übernimmt und in eine Datenbank einspeist, anschließend sollten die Daten mittels Grafana visualisiert werden.

Nach der ersten Bestandsaufnahme der vorhandenen Steuerung und Zählerstellen wurde schnell klar, dass dies so einfach nicht funktionieren würde. Also wurde die alte Steuerung und Infrastruktur in Rente geschickt und durch eine eigene ersetzt. Mit neuen Messstellen, die ihre Werte über einen Bus zur Verfügung stellen. Damit die Daten vom Bus in die Datenbank kommen, wurde eine eigene Schnittstelle in Python programmiert.

Anschließend werden die Daten mit PHP aus der Datenbank geholt und über eine einfache API der Webseite zur Verfügung gestellt. Die Webseite holt die Daten mit Hilfe von JavaScript von der API und stellt sie dann grafisch auf dem Dashboard dar. Zur Steuerung und Überwachung wurde ein gesicherter Admin-Bereich auf der Webseite eingerichtet. Um das Dashboard mit verschiedenen Werten interessanter zu gestalten, wurde die EST Wetterstation mit der Datenbank und der Webseite verbunden. Außerdem werden über die StromGedacht API und über die Bundes API SMARD Daten zur Netzauslastung und zum deutschen Strommix angezeigt. Ich freue mich, Ihnen auf den folgenden Seiten meine Technikerarbeit näher zu erläutern.

Vorab ein paar Eckdaten zur Arbeit:

- 15 Messstellen
- 1400 Potenzielle Messwerte
- 2 Datenbanken
- 3700 Zeilen Code (Python, PHP, JavaScript, HTML, CSS)

Project description

After an event at EST, which focused on the problems and solutions of the current energy turnaround, we considered in a small circle how EST can also set a good example. We quickly came to the conclusion that you can only save energy if you know in the first place how and where this energy is consumed.

For more than 20 years, EST has had a so-called energy mirror, which hangs in the entrance area. The consumption values such as electricity, water, gas and solar are displayed there. Unfortunately, this is now no longer functional. The displays are defective and values such as water and gas are completely missing.

So the foundation stone for this technician work was laid: to replace the old energy mirror with a dashboard in the form of a website. This should be done by a Raspberry Pi, which takes the data from the old control and feeds it into a database, then the data should be visualized using Grafana.

After the initial inventory of the existing control and metering points, it quickly became clear that this would not work so easily. So the old control system and infrastructure were retired and replaced by one of their own. With new metering points that provide their values via a bus. To get the data from the bus into the database, a custom interface was programmed in Python. Then the data is fetched from the database using PHP and made available to the website via a simple API. The website fetches the data from the API using JavaScript and then displays it graphically on the dashboard. A secure admin area was set up on the website for control and monitoring. To make the dashboard more interesting with different values, the EST weather station was connected to the database and the website. In addition, data on grid utilization and the German electricity mix are displayed via the StromGedacht API and via the Bundes API SMARD. I am happy to explain my technical work in more detail on the following pages.

First of all, a few key data about the work:

- 15 measuring points
- 1400 potential measured values
- 2 databases
- 3700 lines of code (Python, PHP, JavaScript, HTML, CSS)

Inhaltsverzeichnis

1. Der Alte Energiespiegel.....	1
1.1 Hersteller.....	1
1.2 Aufbau.....	1
1.3 Probleme.....	3
1.4 Lösungsansatz.....	3
2. Hardware.....	4
2.1 Vorhandene Hardware.....	4
2.2 Neue Hardware.....	4
2.3 Topologie.....	7
2.4 Messbereiche.....	8
3. M-Bus.....	10
3.1 Unterschied M-Bus/Modbus.....	10
4. M-Bus Protokoll.....	11
4.1 Voraussetzung.....	11
4.2 Frames.....	12
4.3 Felder.....	13
4.4 Telegramme.....	13
4.5 Informationsblock.....	15
4.6 Ablauf der Kommunikation.....	16
5. M-Bus Schnittstell.....	17
5.1 Funktionsweise des Programms.....	18
6. Datenbank.....	23
6.1 Topologie.....	23
6.2 Tabellen.....	24
6.3 Views.....	28
7. Webseite.....	29
7.1 Dashboard.....	29
7.2 Unterseiten.....	32
7.3 Admin-Bereich.....	34
7.4 ChartJs.....	35
7.5 Daten Abrufen.....	36
7.6 Externe Quellen.....	37
8. Wetterstation.....	39
9. Sicherheit.....	40
10. Optimierungen.....	40
11. Fazit.....	41

1. Der Alte Energiespiegel

1.1 Hersteller

Der Alte Energiespiegel ist Teil eines Energiemanagementsystems von Econergy, einer Tochtergesellschaft der Econcern Holding Gruppe. Econergy war auf die Vermarktung von Produkten und Dienstleistungen im Bereich Energie und Energieeinsparung spezialisiert. Econcern meldete Ende 2009 Insolvenz an, nachdem die Suche nach einem Investor erfolglos geblieben war.¹

1.2 Aufbau

Kern des Systems ist der Energiemonitor, eine Steuerung, die die Impulswerte von den Zählern (Strom, Wasser, Gas) empfängt, daraus die Verbräuche errechnet und intern in einem Speicher ablegt. Die berechneten Werte können mittels Modem über das Internet an eine Webplattform gesendet werden, wo sie dann ausgewertet werden. Die Hauptfunktion ist jedoch die Weiterleitung der Daten an die eigentliche Energieanzeige, die aus mehreren Anzeigeeinheiten besteht und die erfassten Werte anzeigt. Es ist auch möglich, die aufsummierten Stundenwerte für die einzelnen Medien als Excel-Datei zu exportieren.

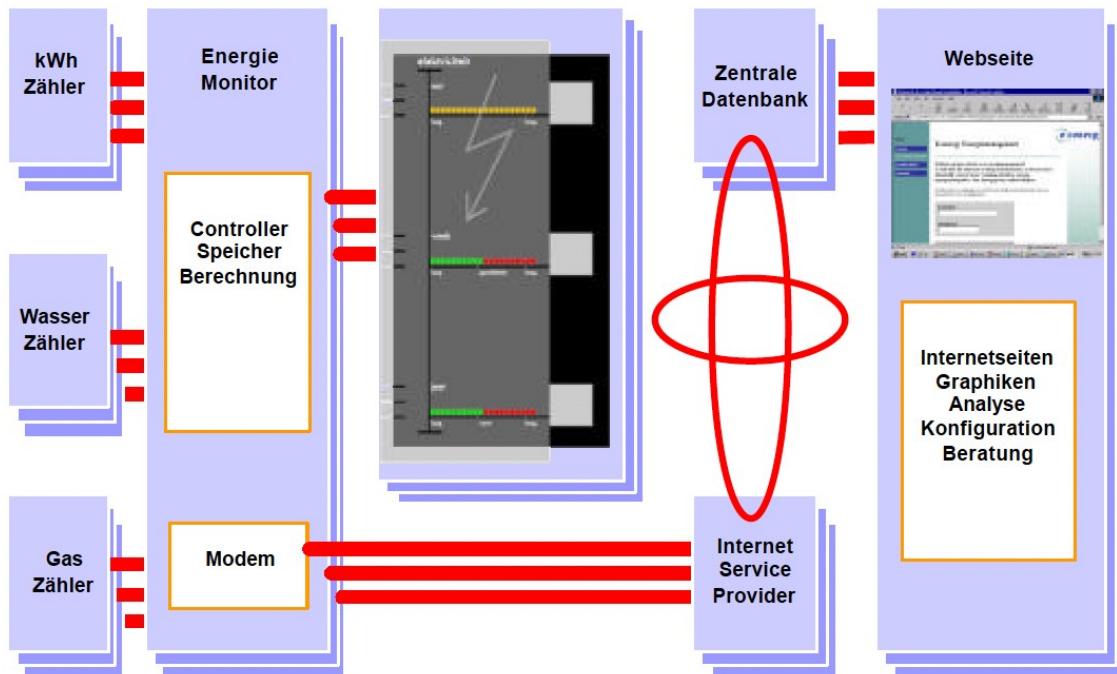


Abbildung 1: Quelle: BROSCHUERE_ENERGIE_MANAGEME S. 2

1 Quelle: <https://en.wikipedia.org/wiki/Econcern>



Abbildung 2: Quelle: BROSCHEURE_ENERGIE_MANAGEME S. 3

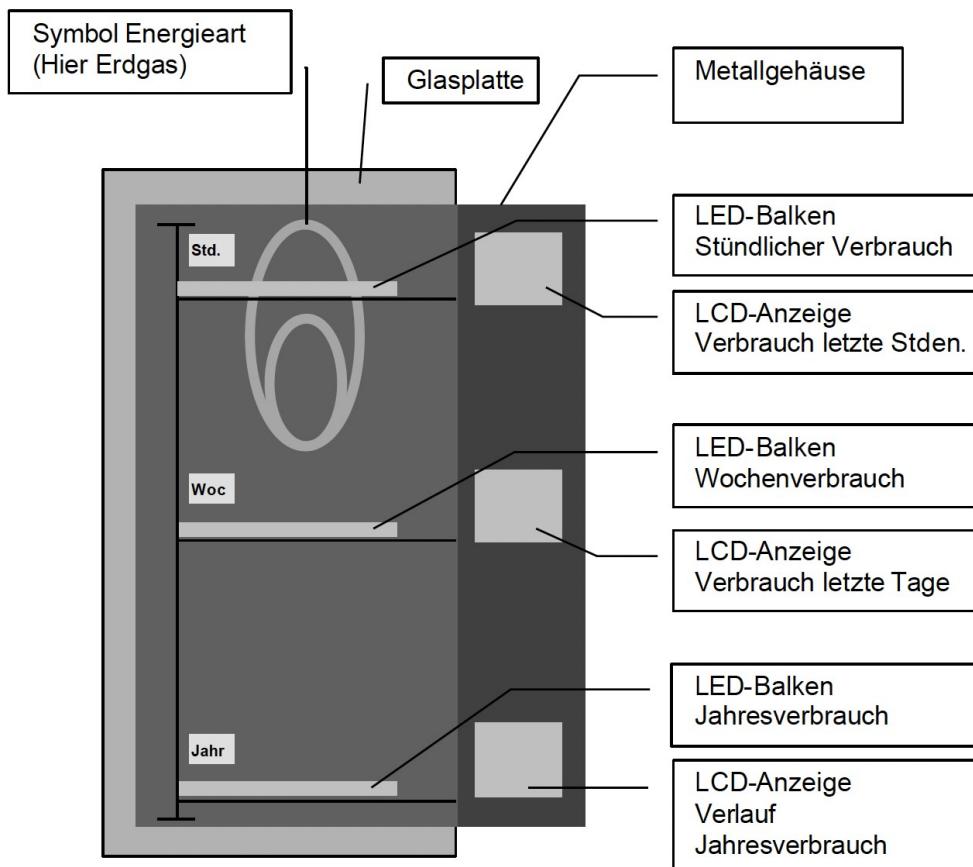


Abbildung 3: HANDBUCH_ENERGIESPIEGEL_V30 S. 5

1.3 Probleme

Dieses System ist in die Jahre gekommen und hat einige Probleme:

- Kommunikationsprotokoll zwischen EnergieMonitor und EnergieSpiegel unbekannt, keine Dokumentation vorhanden.
- Es können nur Werte in voller Stundenauflösung exportiert werden, z.B. wieviel kWh Strom wurden zwischen 10 und 11 Uhr verbraucht.
- Für die Datenübertragung ist eine Software erforderlich, die Windows 95 voraussetzt.
- Diverse LCD-Anzeigen in den Tafeln sind defekt.
- Durch einfache Erfassung über Impulssignale keine weitere Spezifizierung möglich, z.B. Netzspannung bei Strom oder Durchfluss bei Wasser.
- Nicht modular, keine Möglichkeit neue Zählerstellen anzuschließen.
- Kein Service/Ersatzteile, da Firma inzwischen insolvent.

1.4 Lösungsansatz

Aus den oben genannten Problemen wurde der folgende Lösungsansatz abgeleitet:

- Außerbetriebnahme des alten Systems
- Installation eines zentralen Zählerschrankes, in dem die neue Steuerung untergebracht ist.
- Kommunikation über ein Bussystem.
- Einbau von Unterzählern zur genaueren Erfassung der Verbräuche, z.B. im Alt- und Neubau.
- Bus mit einem Raspberry-Pi auslesen und Daten an eine Maria-Db Datenbank senden.
- Einrichtung eines Web-basierten Dachbords.
- System soll modular bleiben, jederzeit erweiterbar mit neuen Zählern.
- Einfache Bedienung, insbesondere beim Anlegen neuer Zähler.

2. Hardware

2.1 Vorhandene Hardware

Einige wenige Komponenten konnten aus dem alten System übernommen werden.

Wasserzähler

Die Wasserzähler wurden vor Jahren erneuert und verfügen neben dem Impulsausgang auch über eine M-Bus Schnittstelle.

Gaszähler

Der Gaszähler kann nicht so einfach durch einen intelligenten Zähler ersetzt werden. Er verfügt jedoch über einen Impulsausgang.

Stromzähler

Der Hauptstromzähler verfügt zwar über eine bidirektionale optische Kommunikationsschnittstelle, über die jedoch nur die Zählerstände abgefragt werden können. Auch dieser Zähler ist nicht ohne weiteres austauschbar, verfügt aber wie der Gaszähler über einen Impulsausgang.

Photovoltaik

Der Stromzähler zur Erfassung der Solarleistung hat ebenfalls einen Impulsausgang.

2.2 Neue Hardware

Bei der Auswahl der neuen Hardware wurde darauf geachtet, dass diese über M-Bus kommunizieren kann. Da der M-Bus eine wichtige Rolle spielt, wird er im folgenden Abschnitt näher erläutert.

Impulswandler

Um auch Zähler ohne direkte M-Bus Schnittstelle auslesen zu können, wurde ein PadPuls MC4 der Firma Relay installiert. An diesen können bis zu vier S0-impulsfähige Zähler angeschlossen werden. Der PadPuls rechnet dann die empfangenen Impulswerte in die Energiewerte der eingestellten Medien um. Auf der M-Bus Seite sieht der PadPuls wie 4 eigenständige Geräte aus.

Pegelwandler

Um den Bus mit Strom zu versorgen und die Daten über eine einfache RS232 Verbindung auslesen zu können, wird ein sogenannter Pegelwandler benötigt. Verwendet wurde ein PW20 der Firma Relay, dieser kann bis zu 20 M-Bus Geräte mit Strom versorgen. Der PW20 ist sozusagen der Übersetzer zwischen M-Bus und RS232.

Wasserzähler

Um den vorhandenen Gesamtwasserverbrauch in Neu- und Altbau aufzuteilen, wurde ein Unterzähler für den Altbau installiert. Verbaut wurde ein Falcon MJ, dieser kann einfach auf einen vorhandenen M100i Zähler aufgesteckt werden und misst mittels Hallsensor die Umdrehungen des Wasserzählers (Abbildung 4). Neben einem Impulsausgang verfügt dieses Modul auch über eine M-Bus Schnittstelle.

Stromzähler

Um den Stromverbrauch genauer aufschlüsseln zu können, wurden insgesamt 9 zusätzliche Stromzähler installiert. Zum Einsatz kamen ABB EQ Energiezähler der B-Serie (B24). Diese Zähler messen die Ströme über Stromwandler. Stromwandler sind Komponenten, die wie eine Zange um eine Stromleitung gehängt werden (Abbildung 6). Sie erzeugen aus einem großen Primärstrom einen kleineren Sekundärstrom im Verhältnis xxx:1 oder xxx:5. Für x muss eine passende Größe gewählt werden, bei Leitungen, die 120 Ampere führen, wäre das z.B. 125:1 oder 125:5. Jeder Zähler benötigt 3 dieser Stromwandler, für jede Phase einen. Es werden also noch 27 Stromwandler benötigt.

Einplatinencomputer

Als Schnittstelle zwischen dem M-Bus und der Datenbank wird ein Einplatinencomputer verwendet. Dieser hat die Aufgabe, die Werte über den Bus von den Zählern abzufragen. Die empfangenen Daten müssen dekodiert und über das Netzwerk an eine virtuelle Serverinstanz gesendet werden. Auf dieser Instanz laufen die Datenbank und das Dashboard. Da es zur Zeit der Technikerarbeit schwierig und teuer war, einen Raspberry-Pi zu bekommen, wurde als Ersatz ein Rock-Pi 4 Model A verwendet. Dieser verfügt über alle Schnittstellen eines Raspberry. Der Rock-Pi verfügt über einen integrierten Speicher von 32GB. Somit ist im Gegensatz zum Raspberry keine SD-Karte notwendig. Es ist auch möglich eine M.2 SSD über einen NVMe Port direkt an das Board anzuschließen.

Sonstiges

Zusätzlich wurde ein Unterverteiler (Abbildung 5) für die neuen Zähler installiert. In diesem Verteiler sind der Pegelwandler und der Einplatinenrechner untergebracht. Da der Pegelwandler eine ungewöhnlich lange Bauform hat und somit nicht auf eine normale Hutschiene passt, wurde ein Gehäuse konstruiert und gedruckt, in dem der Pegelwandler und der Einplatinencomputer Platz finden. Dieses Gehäuse wurde unten am Unterverteiler befestigt (Abbildung7).



Abbildung 4



Abbildung 5

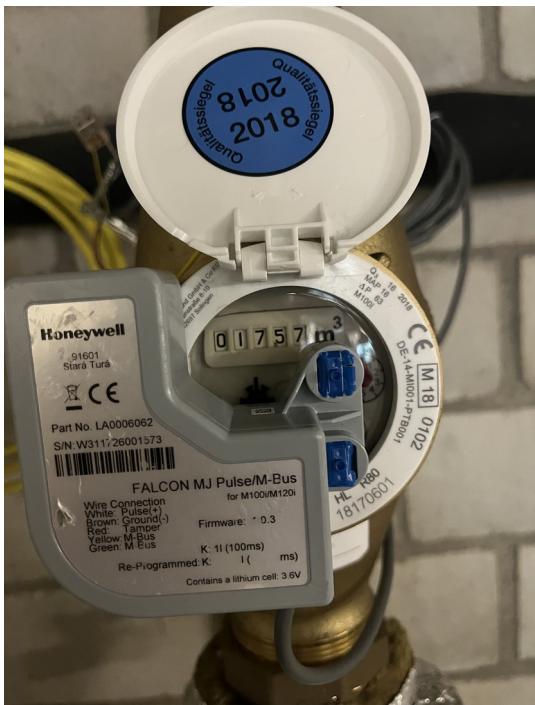


Abbildung 6

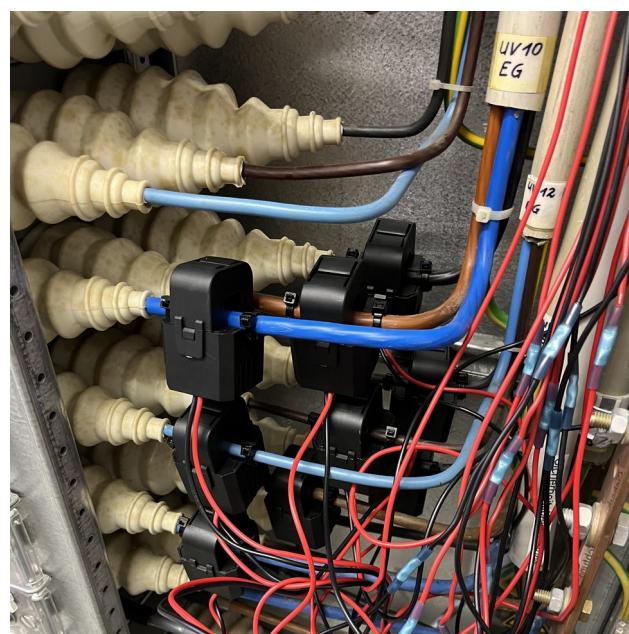
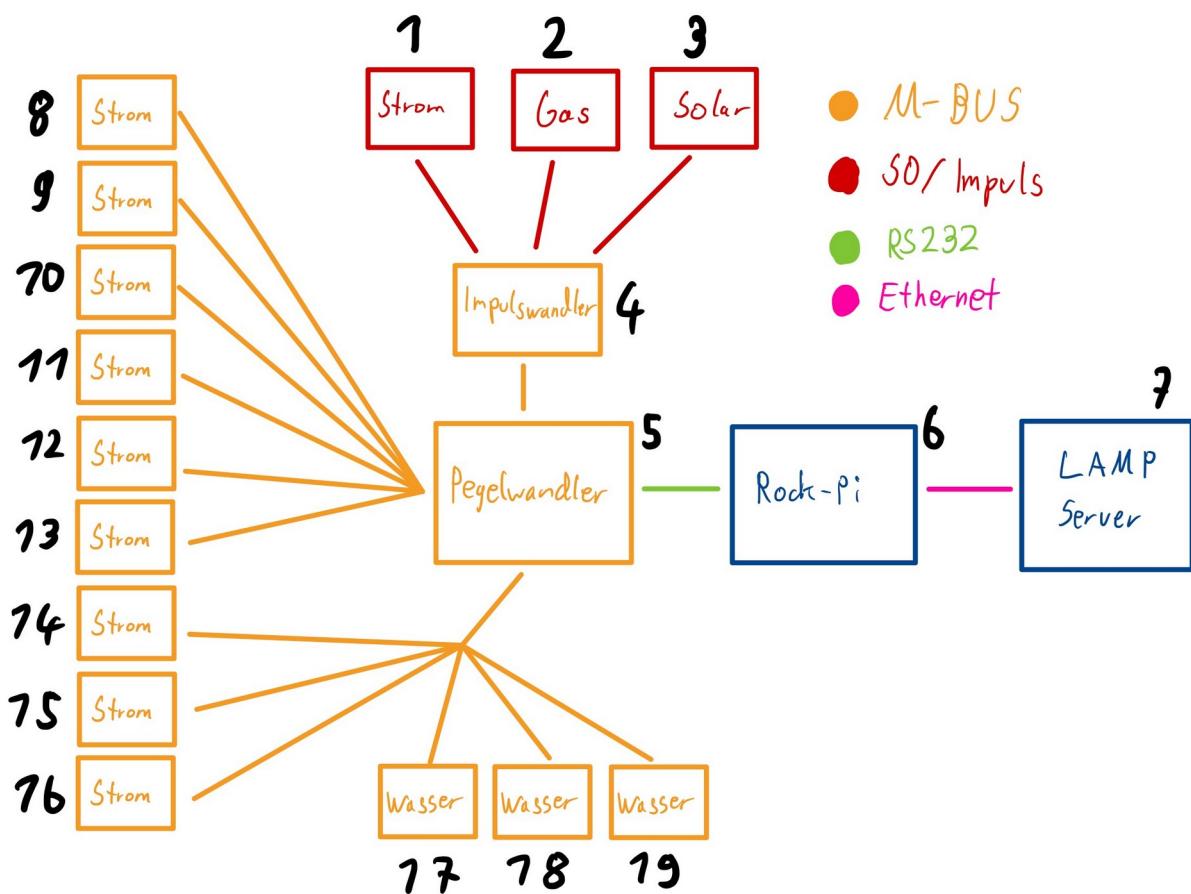


Abbildung 7

2.3 Topologie

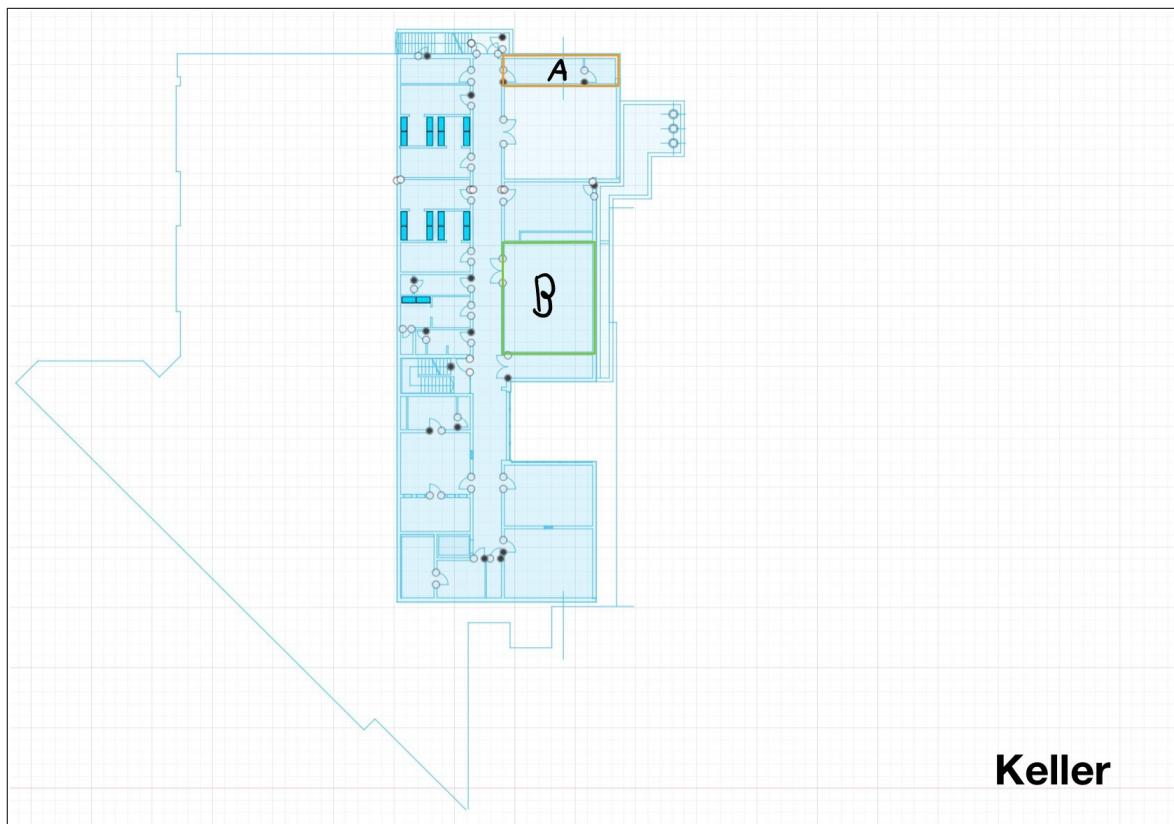
- | | |
|--------------------------------|-------------------------------|
| 1. Hauptstromzähler | 11. Strom Cafeteria |
| 2. Gaszähler | 12. Strom Altbau |
| 3. Solar Stromzähler | 13. Strom Lüftung |
| 4. Impulswandler | 14. Strom Heizung |
| 5. Pegelwandler/M-Bus Master | 15. Strom IT-Räume |
| 6. Rock-Pi | 16. Strom Altbau Obergeschoss |
| 7. LAMP Server | 17. Wasser Gesamt 1 |
| 8. Strom Metallwerkstatt | 18. Wasser Gesamt 2 |
| 9. Strom Elektrowerkstatt UV1 | 19. Wasser Altbau |
| 10. Strom Elektrowerkstatt UV2 | |

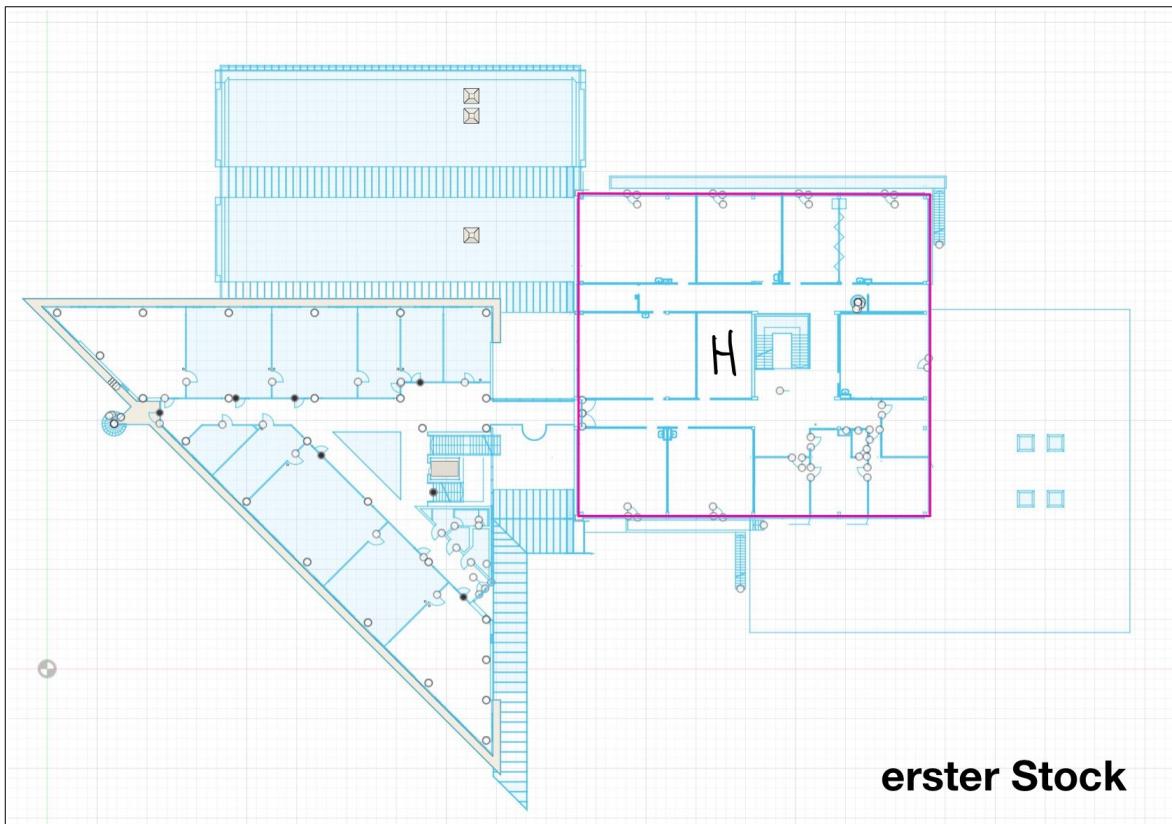
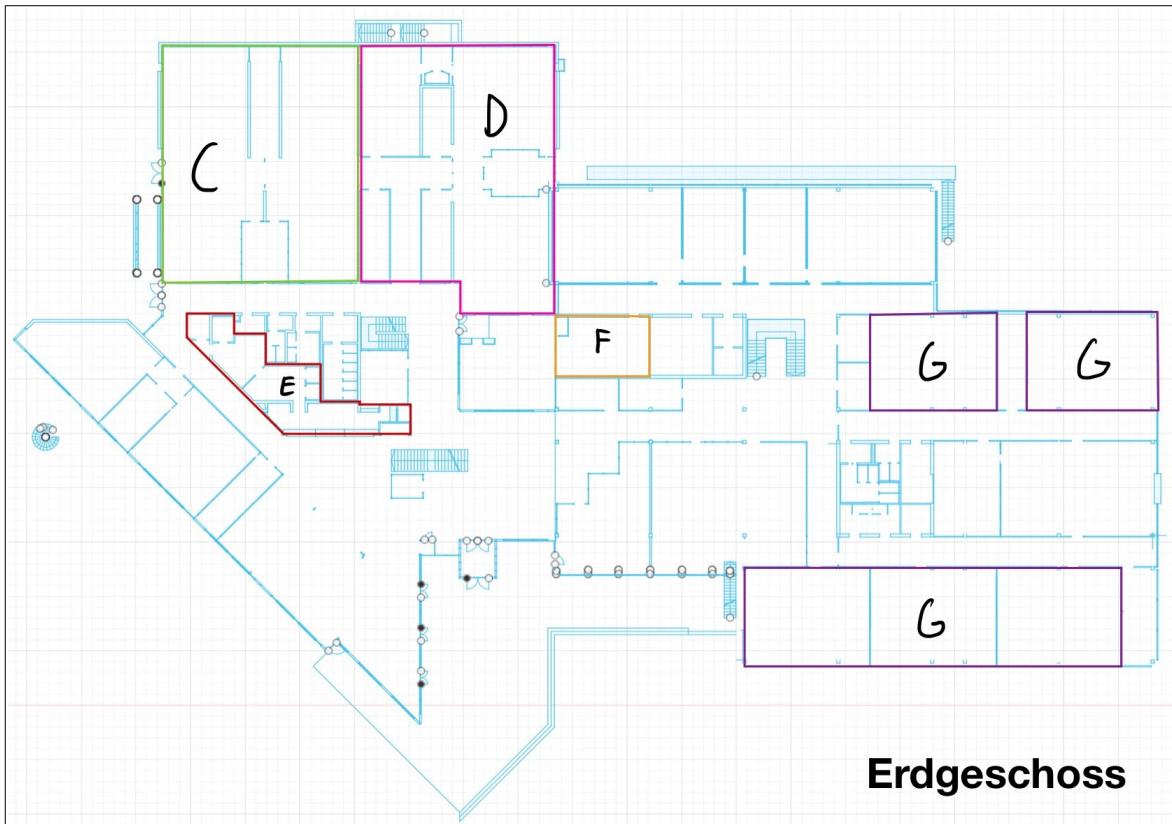


2.4 Messbereiche

Gas, Wasser und Strom werden für das gesamte Gebäude gemessen. Um den Gesamtstromverbrauch aufteilen zu können, werden zusätzlich Messungen in den folgenden Bereichen durchgeführt. Außerdem wird der Wasserverbrauch im Altbau mit einem zusätzlichen Unterzähler gemessen.

- A) Hauptverteiler
- B) Lüftung
- C) Metallwerkstatt
- D) Elektrowerkstatt
- E) Cafeteria
- F) Heizung
- G) IT Räume
- H) Obergeschoss Altbau





3. M-Bus

3.1 Unterschied M-Bus/Modbus

Da beide ähnlich heißen, kann es leicht zu Verwechslungen kommen. So habe ich immer gedacht, dass M-Bus ein Synonym für Modbus ist, aber außer dem ähnlichen Namen haben die beiden nicht viel gemeinsam.

Modbus

Modbus ist streng genommen ein Protokoll. Es kann auf verschiedene Übertragungsmedien angewendet werden. Modbus/RTU kann mit RS232- oder RS485-Systemen verwendet werden. Modbus/TCP kommuniziert über den TCP/IP-Standard. Modbus ist ein in der Industrie weit verbreitetes Kommunikationsprotokoll. Bei der Verwendung von RS485 ist zu beachten, dass die Busenden mit 120 Ohm terminiert werden müssen und eine Linientopologie einzuhalten ist. Durch die weite Verbreitung in der Industrie gibt es viele Stromzähler mit integrierter Modbus-Schnittstelle. Im Gegensatz dazu gibt es nur sehr wenige bis gar keine Wasserzähler mit Modbus.

M-Bus

M-Bus, auch Meter-Bus genannt, wird, wie der Name schon sagt, zur Auslesung von Smart Metern wie Strom-, Gas-, Wasser- und Wärmezählern verwendet. Durch die weite Verbreitung im Smart Meter Bereich gibt es viele Zähler mit integrierter Schnittstelle. Es gibt eine M-Bus Variante, die über einen Zweidrahtbus kommuniziert und eine Variante, die über Funk kommuniziert. Hier unterscheidet sich nur das Übertragungsmedium, das Protokoll ist das gleiche. Der M-Bus basiert auf dem Master-Slave-Modell, der Master sendet eine Anfrage an den Slave und dieser antwortet. Das Faszinierende an diesem Bussystem ist, dass der Master den Slave mit einer Spannungsänderung (0-38V) am Bus anspricht, der Slave antwortet dann mit einer Änderung seiner Stromaufnahme (11mA bis 20mA). Da der Master den Bus und die Geräte mit Strom versorgt, benötigen die Zähler (Slaves) keine zusätzliche Stromversorgung, was besonders bei Wasserzählern nützlich ist, die keine Stromquelle in der Nähe haben. Auch wenn die Slaves nicht vom Master mit Strom versorgt werden, können sie mit Hilfe einer internen Batterie weiterhin Daten sammeln und im internen Speicher ablegen. Die Lebensdauer der Batterien wird mit 10 Jahren angegeben. Außerdem muss bei diesem Busmodell keine Bustopologie beachtet werden, da es keine Terminierung der Enden wie bei einem klassischen Bus gibt. Ein neuer Slave kann an jeder beliebigen Stelle der Busleitung angeschlossen werden. Aufgrund der bereits vorhandenen Wasserzähler mit M-Bus Schnittstelle und der weiten Verbreitung habe ich mich für diese Technologie entschieden.

	M-Bus	Modbus
Architektur	Master-Slave	Master-Slave
Geschwindigkeit	300 bis 9600 Bit/s Standard: 2400 Bit/s	1200 bis 115200 Bit/s Standard: 19200 Bit/s
Max. Leitungslänge	300 Meter	1200 Meter
Max. Teilnehmer	250	32
Bus art	Zweidrige Busleitung, verpolungssicher	RS232/RS485
Bus Topologie	Egal	Linien

4. M-Bus Protokoll

Da das M-Bus-Protokoll insgesamt sehr komplex sein kann, werde ich hier nur auf die Funktionen eingehen, die für die Umsetzung der Technikerarbeit notwendig waren.

4.1 Voraussetzung

Adresse

Jeder am Bus angeschlossene Zähler/Slave benötigt eine eindeutige Adresse, dies kann entweder über die 8-stellige Sekundäradresse oder über die Primäradresse (0-255) erfolgen, wobei nur die Adressen 0 bis 250 zur Adressierung verwendet werden dürfen. Die Adresse 0 ist die Standardadresse im Auslieferungszustand. Die Adressen 251 und 252 sind für zukünftige Anwendungen reserviert. 253 wird bei Anfragen verwendet, um dem Slave mitzuteilen, dass die Identifikation über die Sekundäradressierung erfolgt. Die Adresse 254 kann für Broadcast-Nachrichten an alle Zähler verwendet werden, wobei jeder Zähler mit einer Quittierung antwortet. Die Adresse 255 wird für Broadcast-Nachrichten verwendet, die jedoch nicht von den Slaves quittiert werden. Der Master benötigt keine Adresse.

Baudrate

In jedem Zähler muss eine gültige Baudate konfiguriert werden. Gültige Werte sind 300, 600, 1200, 2400, 4800 und 9600 Baud. Es müssen nicht alle Zähler auf dem Bus die gleiche Geschwindigkeit verwenden, der Master muss nur wissen, mit welcher Geschwindigkeit der angefragte Zähler kommuniziert und stellt seine Geschwindigkeit entsprechend auf die des Zählers ein.

4.2 Frames

Es gibt insgesamt 4 verschiedene Frame arten:

Single Character

Es handelt sich um ein Ein-Byte-Feld mit dem Wert 0xe5. Es wird als Quittierung verwendet. Auch ACK genannt.

Short Frame

Der Frame wird verwendet, um z.B. Zähler über ein SND_NKE auszuwählen. Der Frame hat eine feste Länge von 5 Byte.

Control Frame

Der Control Frame wird verwendet, um Konfigurationsparameter an den Zähler zu senden, z. B. über ein SND_UD, wobei das CI-Feld die neue Baudrate angibt. Der Frame hat eine feste Länge von 9 Byte.

Long Frame

Der Long Frame wird verwendet, um Werte vom Slave zum Master mit RSP_UD zu senden. Der Frame hat eine Mindestlänge von 10 Byte und kann insgesamt 252 Byte Nutzdaten aufnehmen.

Single Character	Short Frame	Control Frame	Long Frame
0xe5	Start 0x10	Start 0x68	Start 0x68
	C Feld	L Feld = 3	L Feld
	A Feld	L Feld = 3	L Feld
	Prüfsumme	Start 0x68	Start 0x68
	Stop 0x16	C Feld	C Feld
		A Feld	A Feld
		CI Feld	CI Feld
		Prüfsumme	Nutzdaten
		Stop 0x16	Prüfsumme
			Stop 0x16

4.3 Felder

A Feld

Dient zur Angabe der Adresse des Zählers. Wenn 253 gesetzt ist, wird die Sekundäradresse anstelle der Primäradresse zur Identifikation verwendet. 254 und 255 sind Broadcast-Adressen.

C Feld

Dient als Kontrollfeld und enthält Informationen über die Flussrichtung und den Nachrichtentyp, z. B. SND_NKE, SND_UD, REQ_UD2, REQ_UD1 oder RSP_UD.

CI Feld

Das Kontrollinformationsfeld enthält Informationen über die Art und Sequenz der Daten.

Prüfsumme

Wird verwendet, um die Vollständigkeit der gesendeten Frames zu überprüfen. Die Prüfsumme wird vom C Feld bis zum Feld vor der Prüfsumme berechnet.

4.4 Telegramme

SND_NKE

Mit diesem Telegramm wird ein Zähler angewählt. Der Zähler bleibt solange angewählt, bis er über den Bus Daten empfängt, die nicht für ihn bestimmt sind. Für die Übertragung wird ein Short Frame verwendet, der Zähler antwortet bei erfolgreichem Empfang mit einem ACK.

SND_UD

Wird verwendet, um einzelne Konfigurationsparameter an den Zähler zu senden. Der Zähler antwortet bei erfolgreichem Empfang mit einer Bestätigung in Form eines ACK.

REQ_UD2

Mit diesem Telegramm werden Nutzdaten vom Zähler angefordert. Für die Übertragung wird ein Short Frame verwendet. Bei erfolgreichem Empfang antwortet der Zähler mit einem RSP_UD.

RSP_UD

Mit diesem Telegramm sendet der Zähler seine Nutzdaten an den Master.

Beispiel REQ_UD2

Byte Nr.	Größe	Wert in Hex	Beschreibung
1	1	10	Start Zeichen Short Frame
2	1	7B	C-Feld, Auslesedaten übertragen
3	1	0A	A-Feld, Primäradresse, hier Adresse Nr. 10
4	1	85	CS-Feld, Summe aus C und A Feld (0x7B + 0x0A)
5	1	16	End Zeichen

Beispiel RSP_UD

Byte Nr.	Größe	Wert in Hex	Beschreibung
1	1	68	Start Zeichen Long Frame
2	1	XX	L-Feld, je nach Anzahl der Auslesedaten
3	1	XX	L-Feld Wiederholung
4	1	68	Start Zeichen Long Frame Wiederholung
5	1	08	C-Feld, Übertragung von Auslesedaten
6	1	0A	A-Feld, Primäradresse, hier Adresse Nr. 10
7	1	71	CI-Feld, Auslesedaten des M-Bus Moduls
8-11	4	XX	Seriennummer, 8 BCD-Ziffern
12/13	2	4204	Hersteller Kennung: ABB
14	1	XX	Version der Firmware
15	1	07	Medium = Wasser
16	1	XX	Zugriffszähler
17	1	XX	Status/Fehlerflags
18/19	2	00 00	Signatur, immer 00 00
20-XX	xx	xx	Auslesedaten, je nach Anzahl der übertragenen Felder
XX+1	1	xx	CS-Feld, Summe von C-Feld bis Ende Auslesedaten
17	1	16	End Zeichen

4.5 Informationsblock

Innerhalb der Auslesedaten werden die einzelnen Werte in Informationsblöcken (DIB, Data Information Block) gespeichert. Jeder Block beginnt mit einem DIF (Data Information Field), dieses ist ein Byte groß und gibt an, welchen Datentyp der Wert hat. Dem DIF können bis zu 10 DIFE (DIF-Extension) folgen, die ebenfalls jeweils ein Byte groß sind. Diese geben weitere Informationen über den Wert. Nach dem DIFE folgt ein VIF (Value Information Field), das angibt, worum es sich bei dem Wert handelt, z.B. Wirkenergie oder Blindenergie. Nach dem VIF können bis zu 10 VIFE (VIF-Extension) folgen, diese geben nähere Informationen zu dem Wert, z.B. ob es sich um die Wirkleistung für Phase L1, L2 oder L3 handelt. Erst nach dem VIFE folgt der eigentliche Wert.

Aufbau

DIF	DIFE	VIF	VIFE	Wert
1 Byte	0-10 Byte	1 Byte	0-10 Byte	0-n byte

Mögliche Datentypen

Die letzten 4 Bits des DIF-Feldes geben den Datentyp an. Es folgt eine Auswahl der am häufigsten verwendeten Datentypen.

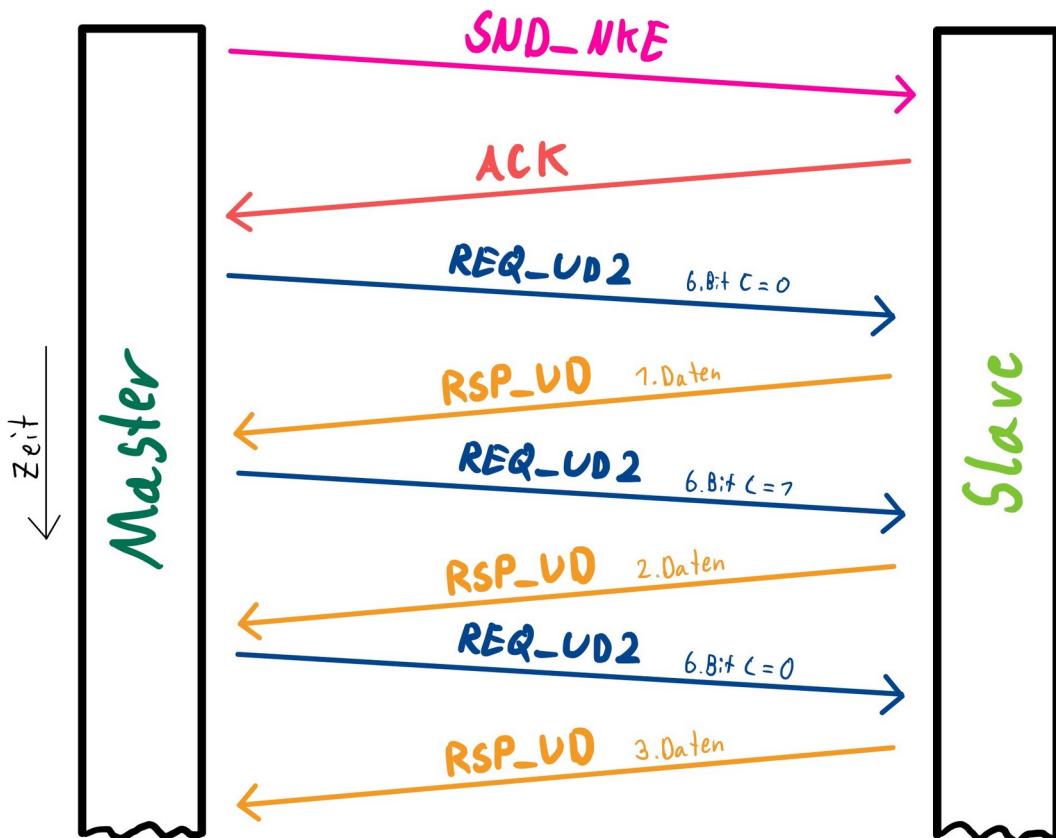
DIF	Anzahl Byte	Codierung (LSB first)
0x0	0	Keine Daten verfügbar
0x1	1	8 Bit-Ganzzahl oder Bit Feld.
0x2	2	16 Bit-Ganzzahl mit Vorzeichen
0x3	3	24 Bit-Ganzzahl mit Vorzeichen
0x4	4	32 Bit-Ganzzahl mit Vorzeichen
0x5	4	Gleitkomma
0x6	6	48 Bit-Ganzzahl mit Vorzeichen
0xC	4	BCD, 8-Ziffern
0xD	Variabel	ASCII Text

4.6 Ablauf der Kommunikation

Es gibt verschiedene Möglichkeiten, Daten vom Zähler zu erhalten. Eine Möglichkeit ist, dem Zähler mitzuteilen, welche Daten man haben möchte. Eine andere Möglichkeit ist dem Zähler mitzuteilen, dass er alle seine Daten senden soll. Letzteres Benutze ich bei meiner Schnittstelle.

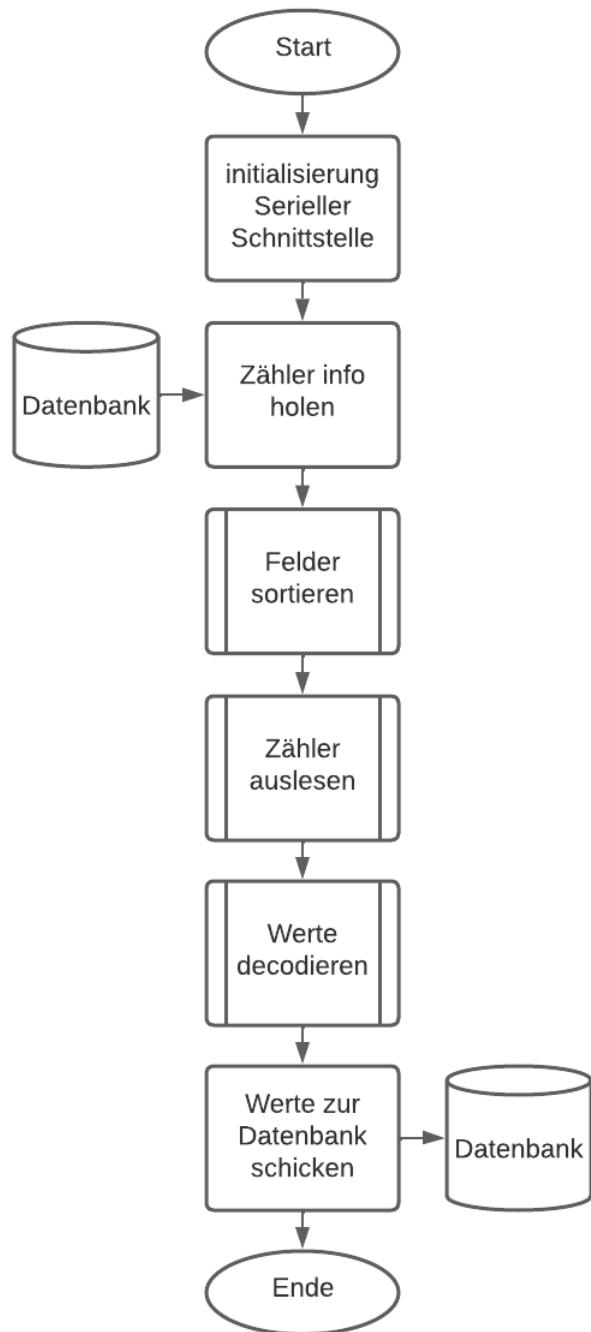
Ablauf

Zuerst wird der Zähler mit einem SND_NKE in Bereitschaft gesetzt. Sobald der Master vom Slave eine Antwort in Form einer ACK erhält, sendet der Master eine Anforderung der Nutzdaten mit Hilfe der Nachricht REQ_UD2. Der Zähler antwortet mit seinen Nutzdaten in Form einer RSP_UD. Da pro Long Frame maximal 252 Byte Nutzdaten übertragen werden können, fordert der Master nach erfolgreichem Empfang der letzten Daten den Zähler mit einem REQ_UD2 auf, neue Daten zu senden. Dabei wird jedoch das 6. Bit im C-Feld gedreht, was dem Zähler anzeigt, dass das letzte Paket erfolgreich angekommen ist und er das nächste Paket senden kann. Dies geschieht solange, bis der Master die Anfrage abbricht oder alle Nutzdaten übertragen wurden. Wurde das 6. Bit nicht gedreht, bedeutet dies, dass der Frame nicht erfolgreich angekommen ist. Zur Korrektur sendet der Zähler den letzten Frame erneut.



5. M-Bus Schnittstelle

Als Bindeglied zwischen M-Bus und Datenbank wurde ein Python-Programm entwickelt. Dieses Programm läuft auf einem Rock-Pi Einplatinencomputer. Dieser Computer ist über ein serielles Kabel mit dem M-Bus Pegelwandler verbunden. Auf dem Rock-Pi läuft eine einfache Ubuntu-Instanz. Diese Instanz startet das Python-Programm jede Minute mit Hilfe eines Cron-Jobs. Das Programm besteht aus zwei Teilen, die abfrage.py holt die benötigten Abfragefelder aus der Datenbank und sortiert sie. Danach wird ein Objekt der Klasse mbus_master.py erzeugt, welches das eigentliche Hauptprogramm darstellt. Dieses erhält die benötigten Werte und fragt diese beim Zähler ab, anschließend werden die Werte wieder an das Programm abfrage.py zurückgegeben. Dieses wiederum sendet die Daten an die Datenbank. Bei der Entwicklung des Programms mbus_master.py wurde bewusst auf die Verwendung einer M-Bus Bibliothek verzichtet. Die mbus_master.py kann unabhängig von der abfrage.py verwendet werden, um z.B. zu Diagnosezwecken den M-Bus direkt über die Konsole mittels eines Python-Skripts anzusprechen. Auf der rechten Seite ist der vereinfachte Programmablauf dargestellt. Auf den folgenden Seiten wird das Programm im Detail erläutert. Zur besseren Veranschaulichung wurden Flussdiagramme anstelle von Code verwendet, der Quellcode kann im Anhang eingesehen werden.



5.1 Funktionsweise des Programms

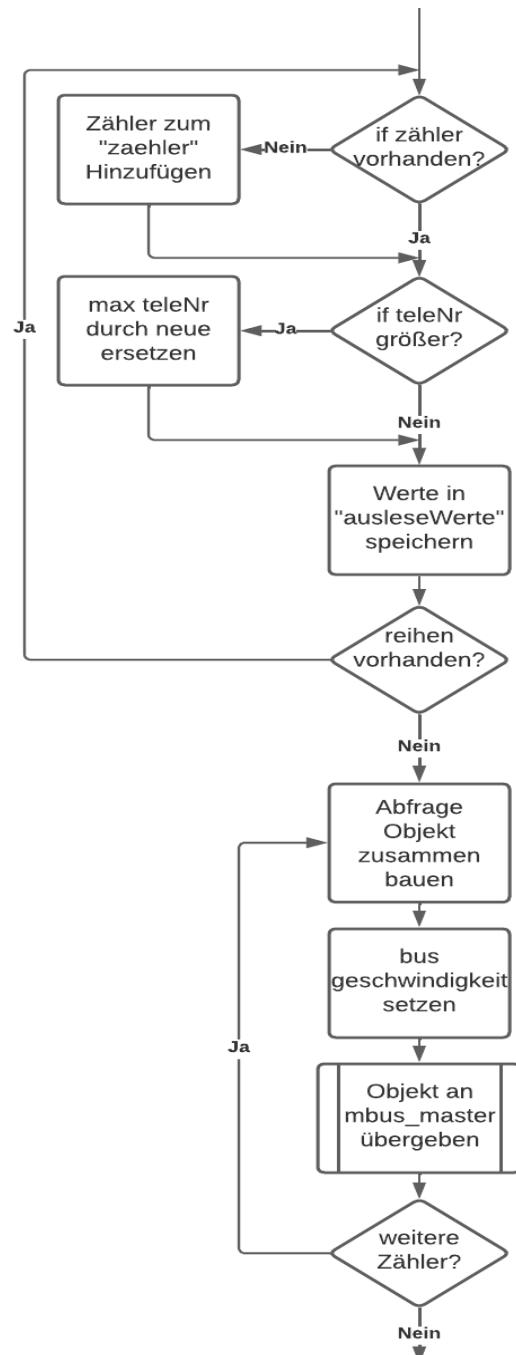
Vorbereitung

Zunächst werden die erforderlichen Informationen aus der Datenbank abgerufen. Jede Zeile, die von der Tabelle empfangen wird, enthält alle Informationen, die für die Abfrage eines Wertes erforderlich sind.

WertNr	Adresse	Bcd	StartFeld	EndFeld	TeleNr	Multiplikator	Speicherung	Baudrate
36	20	0	34	35	3	0.001000	0	9600

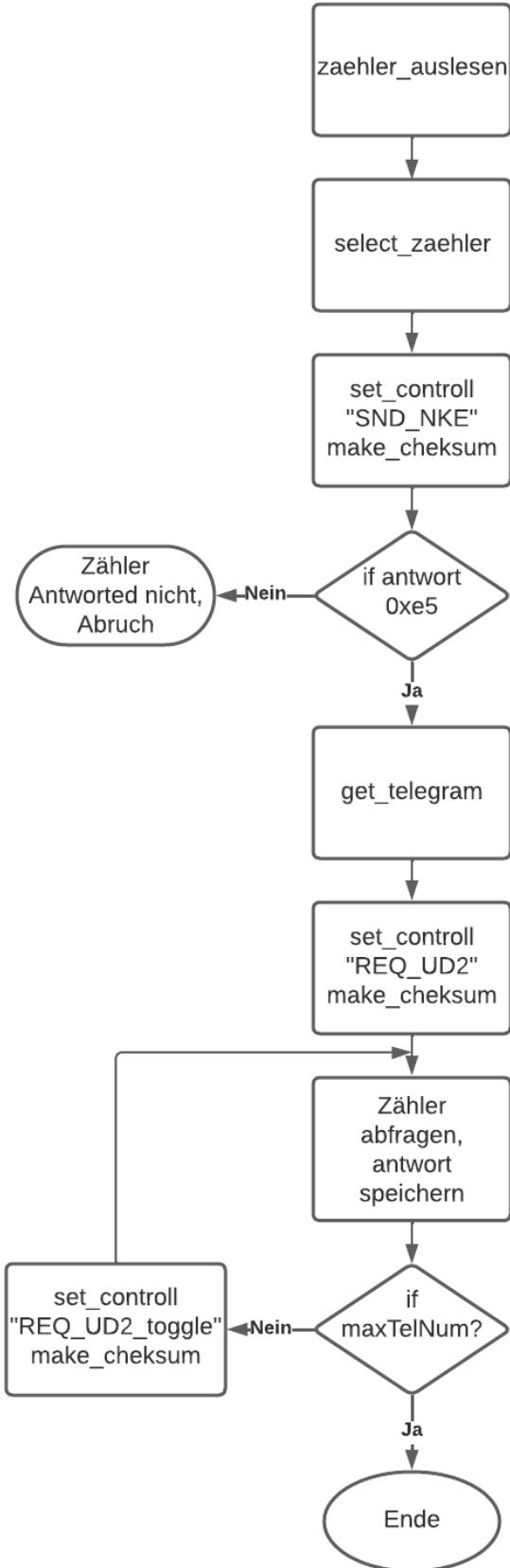
Für jede empfangene Zeile wird die Adresse und die TeleNr in einem zweidimensionalen Array gespeichert, so dass die Adresse und die zugehörige TeleNr in Beziehung stehen. Der Rest der Zeile wird in einem weiteren zweidimensionalen Array gespeichert, wobei die zweite Dimension ein assoziatives Array ist. Für jede neu empfangene Zeile wird geprüft, ob die Adresse bereits im Array vorhanden ist, und wenn nicht, wird sie hinzugefügt. Außerdem wird geprüft, ob die neu empfangene TeleNr größer ist als die gespeicherte TeleNr aus dem Array, wenn ja, wird die vorhandene TeleNr ersetzt. Dadurch wird sichergestellt, dass jeder Zähler mit seiner maximalen TeleNr nur einmal vorhanden ist. Die restlichen Werte werden als neue Schicht in das zweite Array eingefügt. Nachdem alle Zeilen durchlaufen wurden, erhält man zwei Arrays, eines mit den Abfragefeldern und eines mit den Adressen und TeleNr.

Im nächsten Schritt wird die serielle Geschwindigkeit für den abzufragenden Zähler eingestellt, danach wird für jeden Zähler ein Objekt der Klasse mbus_master erzeugt, wobei die Adresse, TeleNr und das PySerial Objekt übergeben werden. Danach wird das eigentliche Auslesen mit zaehler_auslesen gestartet. Der Vorgang wird solange wiederholt, bis das Array mit den Adressen durchlaufen und somit jeder benötigte Zähler einmal ausgelesen wurde, dabei werden die empfangenen Daten sortiert nach ihrer TeleNr im zugehörigen Objekt gespeichert.



Zähler abfragen

Nach Übergabe des Objektes und Start des Auslesemechanismus mit zaehler_auslesen wird die Funktion select_zahler aufgerufen. Innerhalb dieser Funktion wird set_controll mit dem Parameter „SND_NKE“ aufgerufen. Dadurch wird ein Control-Frame mit den übergebenen Parametern gebaut, einschließlich der Berechnung der Prüfsumme. Nachdem der Frame zusammen gebaut ist, wird er auf den Bus gesendet, es wird abgewartet, ob eine Antwort in Form eines „ACK“ vom Zähler kommt, wenn nicht, wird ein „False“ zurückgegeben und die Abfrage des Zählers abgebrochen. Wenn ein „ACK“ empfangen wurde, wird ein „True“ zurückgegeben und das Programm fährt mit der Funktion get_telegram fort. Innerhalb dieser Funktion wird erneut die Funktion SET_CONTROL aufgerufen, um einen „REQ_UD2“ Kontrollrahmen zu erhalten. Dieses Telegramm wird auf den Bus gesendet und die empfangenen Daten werden als String in einem Array gespeichert. Danach wird überprüft, ob noch weitere Pakete für diesen Zähler benötigt werden, dies geschieht mit dem maximalen TeleNr Wert, der bei der Objekterzeugung übergeben wurde. Wenn noch mehr Pakete benötigt werden, wird die Funktion set_control aufgerufen, um ein „REQ_UD2“-Frame mit invertierten 6. Bit im C-Feld zu erhalten, das bewirkt, dass nach dem Senden des Frames an den Zähler dieser mit dem nächsten Paket antwortet, dieses wird wieder als String in das Array eingefügt. Dies geschieht solange, bis alle erforderlichen Pakete empfangen wurden.

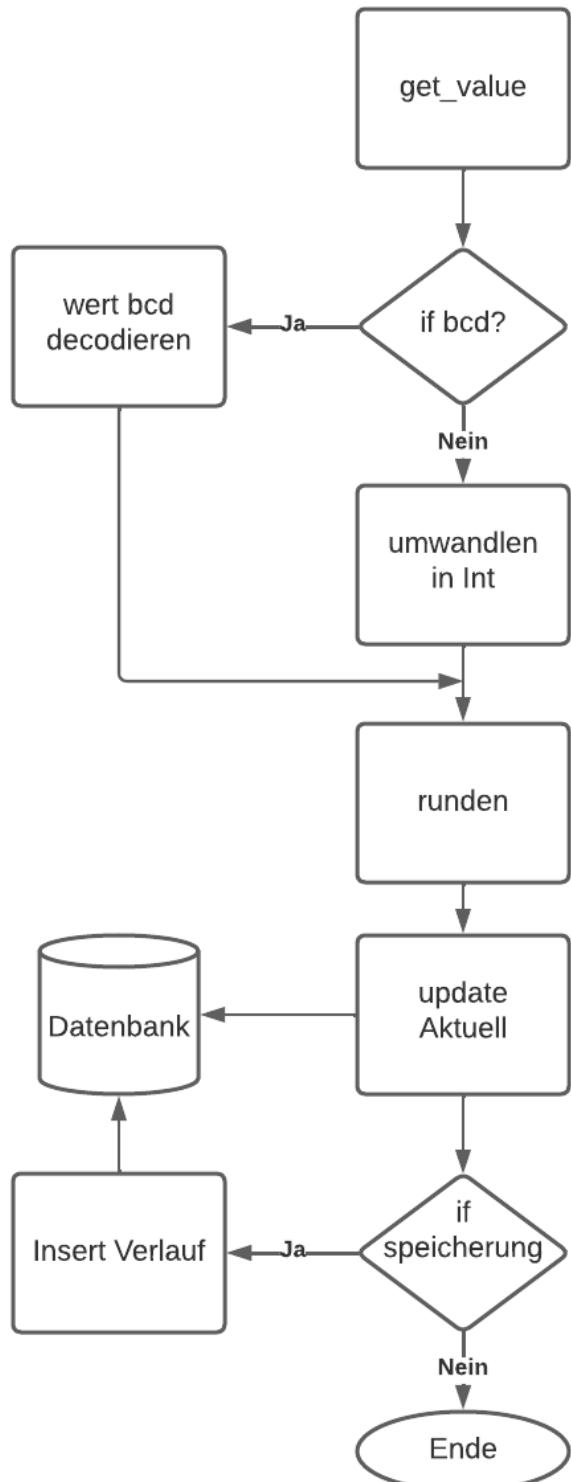


Werte decodieren und Hochladen

Nachdem alle Zähler ausgelesen wurden, werden die Werte aus dem Paket mit Hilfe des assoziativen Arrays dekodiert. Zuerst wird das Objekt, in dem sich die Werte befinden, mit Hilfe der Adresse gefunden, dann wird die Funktion `get_value` mit den Parametern `TeleNr`, `Startfeld`, `Endfeld`, `Multiplikator` und `BCD` aufgerufen.

Innerhalb der Funktion `get_value` wird der Rohwert mit Hilfe der Parameter `Start-` und `Endfeld`, die die Anzahl der Bytes angeben, in denen sich der Wert befindet, aus dem Paket extrahiert. Dann wird mit BCD geprüft, ob es sich um einen BCD-codierten Wert handelt oder nicht, und wenn der Wert BCD-codiert ist, wird er decodiert. Wenn nicht, handelt es sich um einen numerischen Wert. Dann wird der Wert, egal ob BCD oder nicht, mit dem angegebenen Multiplikator multipliziert und gerundet. Das Ergebnis wird zurückgegeben. Der zurückgegebene Wert wird zusammen mit der `WertNr` und der aktuellen Zeit/Datum an die `Aktuell`-Tabelle gesendet, wo der alte Wert durch den neuen Wert ersetzt wird. Wenn die Speicherung auf „True“ gesetzt ist, wird der Wert, zusammen mit der `WertNr` und der aktuellen Zeit/Datum, zusätzlich in die Verlaufstabelle eingefügt. Dadurch wird die Historie des Wertes aufgezeichnet. Diese Schritte werden nun für jeden Wert im assoziativen Array wiederholt.

Damit ist die Abfrage beendet und alle Werte sind in der Datenbank gespeichert. Der Rest des Programms `abfrage.py` befasst sich mit der Bearbeitung/Berechnung spezieller Werte.

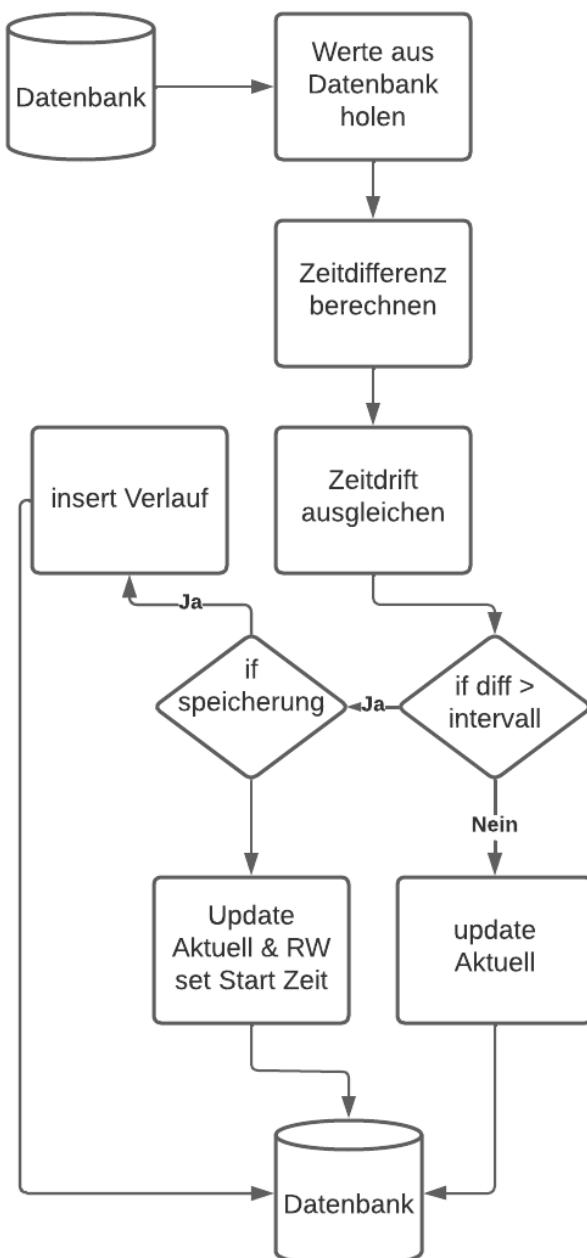


Verbrauchswerte Berechnen

Mit diesem Programmabschnitt können Stunden-, Tages-, Wochen-, Monats- und Jahreswerte ermittelt werden. Zu Beginn werden folgende Werte aus der Datenbank geholt.

ID	EingangsWertNr	AusgangsWertNr	ZeitstempelAnfang	Anfangsstand	Intervall	ZeitType	Wert	Speicherung
23	43	67	2023-06-21 00:00:00	3964.180	1	DAY	4042.82	1

Anschließend wird die Differenz anhand des Feldes ZeitstempelAnfang in Verbindung mit der aktuellen Zeit berechnet. Da nicht sichergestellt werden kann, dass die Daten wirklich zeitgenau in die Datenbank eingetragen werden, kann es vorkommen, dass aus 20:00:00 erst 20:00:53, dann 20:01:32 usw. wird. Um dies zu verhindern, wird die aktuelle Zeit je nach TimeType angepasst, z.B. bei einem TimeType von „hour“ und einer Zeit von „2023-06-21 20:01:54“ werden die Sekunden und Minuten auf 0 gesetzt, so dass „2023-06-21 20:00:00“ herauskommt. Dies hat zur Folge, dass der Zeitstempel (ZeitstempelAnfang) immer auf den ganzzahligen Wert des zugehörigen Zeittypfeldes gerundet wird. Ist die Zeitdifferenz größer als das eingestellte Intervall (Summe aus Intervall und ZeitType, z.B. 2 DAY = 48 Stunden), wird der neue Wert zusammen mit WertNr und ZeitstempelAnfang in der History-Tabelle gespeichert. Der Wert wird berechnet, indem der aktuelle Wert mit dem Anfangswert subtrahiert wird. Anschließend wird die Tabelle ZählerstandRechner durch den neuen Anfangsstand (aktueller Wert der Eingangsnummer, in diesem Fall 4042,82) ersetzt und der ZeitstempelAnfang durch den korrigierten Zeitstempel ersetzt. Wenn die Zeitdifferenz kleiner als das Intervall ist, wird nur der Wert und der Zeitstempel in der Aktuell-Tabelle aktualisiert.

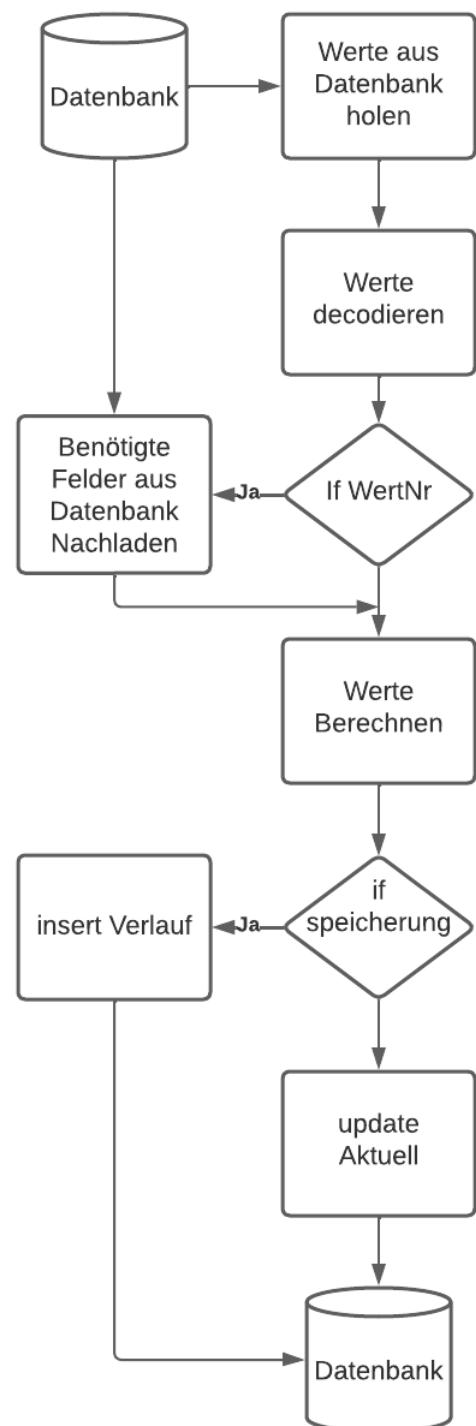


Berechnungen

Um alle Arten von Berechnungen automatisch durchführen zu können, wurde eine spezielle Tabelle erstellt.

id	EingangsWertNr	berechnung	AusgangsWertNr
1	39	{"werte": [66, 47, 81], "isNr": [true, true, true], "operant": ["-", "-", "-"], "="]}	80

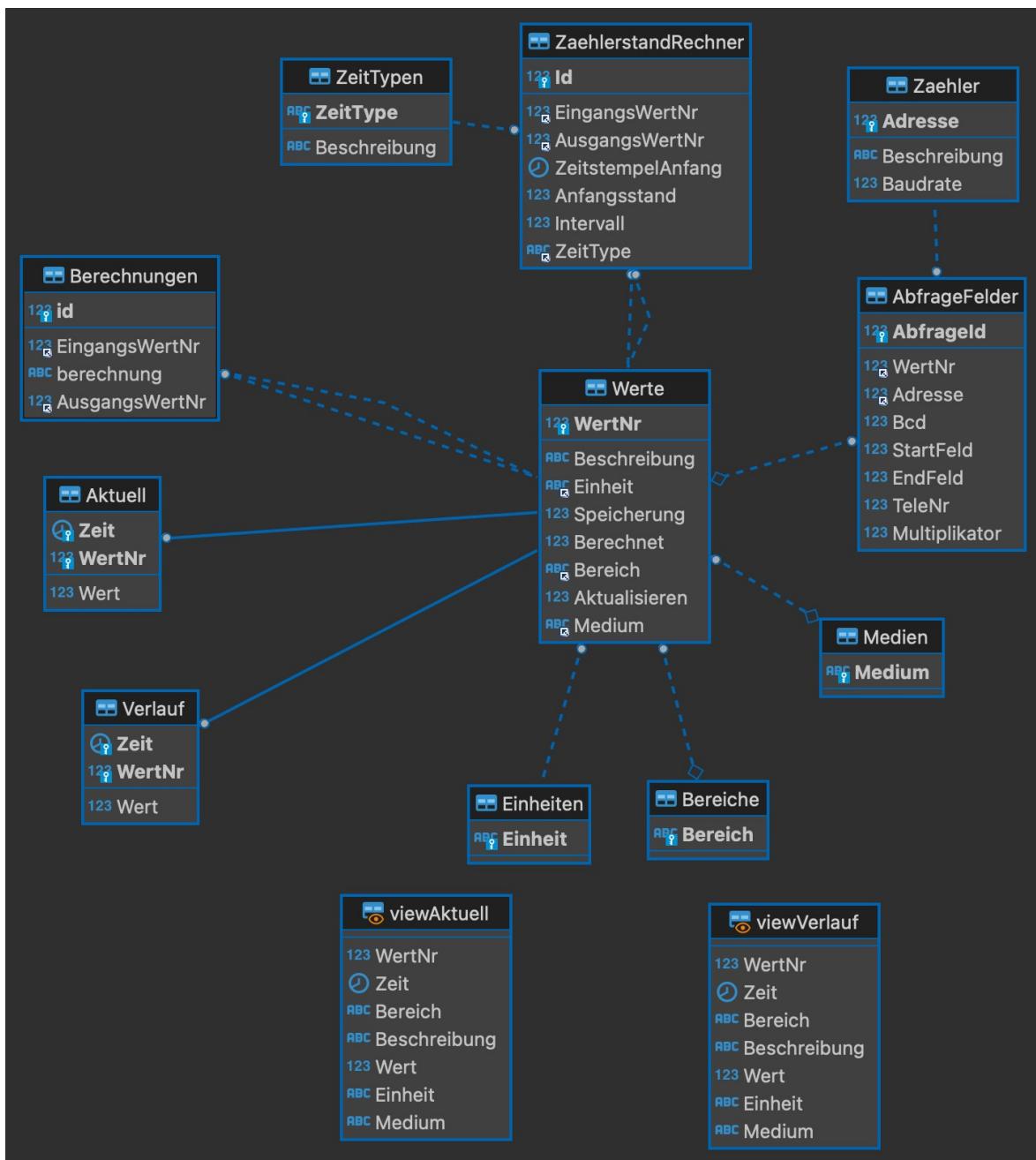
Hier werden alle für die Berechnung relevanten Informationen gespeichert. „EingangsWertNr“ gibt an, welcher Anfangswert aus der Aktuell-Tabelle geholt werden soll. „AusgangsWertNr“ gibt an, unter welcher WertNr der Wert gespeichert werden soll. Das Feld „Berechnung“ ist vom Typ „longtext“ und enthält einen Json String. „werte“ gibt entweder eine „WertNr“ oder eine beliebige Zahl an. „isNr“ gibt an, ob es sich bei „werte“ um „WertNr“ oder um normale Zahlen handelt. „Operant“ gibt an, welche Operation zwischen den einzelnen „werte“ durchgeführt werden soll. Dabei gibt die erste Stelle die Operation zwischen „EingangsWertNr“ und dem ersten Wert an. Diese Information wird vom Programm aus der Datenbank gelesen. Danach werden alle benötigten „WertNr“-Felder aus der Aktuell-Tabelle geholt. Anschließend wird die Berechnung durchgeführt. Am Ende wird das Ergebnis zusammen mit der aktuellen Zeit in die Aktuell-Tabelle geladen. Wenn das Speicherfeld aktiviert ist, wird der Wert zusätzlich mit dem aktuellen Zeitstempel in die Historientabelle hochgeladen.



6. Datenbank

Als DBMS (Datenbankmanagementsystem) habe ich mich für MariaDB entschieden. Diese läuft zusammen mit einem Apache Server auf einer virtuellen Ubuntu Server Instanz. Dabei handelt es sich um einen sogenannten LAMP-Server. L = Linux (Betriebssystem), A = Apache (Webserver), M = MariaDB/MySQL (Datenbank) und P = PHP (serverseitige Programmiersprache).

6.1 Topologie



6.2 Tabellen

Das Design der Datenbank wurde so gewählt, dass möglichst viele Werte durch Tabellen vorgegeben werden, z.B. Bereiche, Medien, Einheit. Dies soll verhindern, dass versehentlich Werte dem Bereich altbau statt Altbau zugeordnet werden. Dieses Design verhindert auch, dass Werte einem Zähler zugewiesen werden, der gar nicht existiert.

Werte

Herzstück der Datenbank ist die „Werte“ Tabelle.

Field	Type	Null	Key	Default	Extra
WertNr	int(11)	NO	PRI	NULL	auto_increment
Beschreibung	varchar(50)	NO		NULL	
Einheit	varchar(10)	NO	MUL	NULL	
Speicherung	tinyint(1)	NO		NULL	
Berechnet	tinyint(1)	NO		NULL	
Bereich	varchar(50)	NO	MUL	NULL	
Aktualisieren	tinyint(1)	NO		1	
Medium	varchar(10)	NO	MUL	NULL	

Diese Tabelle enthält alle wesentlichen Informationen zu den einzelnen Werten. Jeder Wert erhält eine eindeutige WertNr, mit dieser er in der gesamten Datenbank identifiziert wird. Die Beschreibung gibt an, um was für einen Wert es sich handelt, z.B. Wirkleistung. Als Einheit können nur Werte verwendet werden, die auch in der Einheitentabelle vorhanden sind, z.B. kWh, l/h etc. Speicherung gibt an, ob der Wert auch in die Verlauf-Tabelle geschrieben werden soll. Berechnet gibt an, ob der Wert ausgelesen oder aus anderen Werten berechnet wird. Aktualisieren gibt an, ob der Wert aktualisiert werden soll. Mit dem Feld Bereich wird der Wert einem vordefinierten Bereich zugeordnet, z.B. Altbau, Heizung... Mit Medium wird dem Wert ein Medium wie Wasser, Gas oder Strom zugeordnet, auch hier dürfen nur Werte aus der Medien-Tabelle verwendet werden. Berechnung und Aktualisierung sind wichtige Felder für das M-Bus Programm, da das Programm nur Werte abfragt, die tatsächlich benötigt werden und auf dem Zähler vorhanden sind.

Medien

Field	Type	Null	Key	Default	Extra
Medium	varchar(10)	NO	PRI	NULL	

Die Medientabelle ist eine Tabelle, die dazu dient, vordefinierte Medien (Wasser, Gas, Strom, ...) zur Verfügung zu stellen.

AbfrageFelder

Field	Type	Null	Key	Default	Extra
AbfrageId	int(11)	NO	PRI	NULL	auto_increment
WertNr	int(11)	NO	MUL	NULL	
Adresse	int(11)	NO	MUL	NULL	
Bcd	tinyint(1)	NO		NULL	
StartFeld	int(11)	NO		NULL	
EndFeld	int(11)	NO		NULL	
TeleNr	int(11)	NO		NULL	
Multiplikator	double(10,6)	NO		NULL	

Diese Tabelle enthält Informationen über die Werte die vom Zähler gelesenen werden sollen. Diese Informationen sind der Betriebsanleitung des Zählers zu entnehmen. Auch hier gibt es vordefinierte Werte, d.h. es können nur Werte eingetragen werden, die auch in der Wertetabelle vorhanden sind, und es können nur Adressen ausgewählt werden, die in der Zählertabelle vorhanden sind.

Zaehler

Field	Type	Null	Key	Default	Extra
Adresse	int(11)	NO	PRI	NULL	
Beschreibung	varchar(20)	NO		NULL	
Baudrate	int(11)	NO		2400	

Diese Tabelle enthält Informationen über den Zähler. Die Adresse ist ein Primärschlüssel, der sicherstellt, dass jede Adresse nur einmal existiert. Mit der Beschreibung kann dem Zähler ein Name gegeben werden. Das Feld Baudrate gibt an, mit welcher Busgeschwindigkeit der Zähler angesprochen werden soll.

Bereiche

Field	Type	Null	Key	Default	Extra
Bereich	varchar(50)	NO	PRI	NULL	

Die Bereichstabelle stellt vordefinierte Bereiche (Altbau, Cafeteria, ...) zur Verfügung, um sicherzustellen, dass jeder Wert einem gültigen Bereich zugeordnet werden kann.

Einheiten

Field	Type	Null	Key	Default	Extra
Einheit	varchar(5)	NO	PRI	NULL	

Wie die Medien- und Bereichstabelle liefert auch die Einheitentabelle vordefinierte Werte, z.B. kWh, A, V, etc.

Berechnungen

Field	Type	Null	Key	Default	Extra
<i>id</i>	int(11)	NO	PRI	NULL	
EingangsWertNr	int(11)	NO	MUL	NULL	
berechnung	longtext	NO		NULL	
AusgangsWertNr	int(11)	NO	MUL	NULL	

Diese Tabelle ist nur für die Speicherung von Informationen über wiederkehrende Berechnungen vorgesehen. Die EingangsWertNr und die AusgangsWertNr müssen gültige Werte aus der Wertetabelle sein. Das Berechnungsfeld enthält die eigentliche Rechenoperation in Form eines JSON-Strings.

ZaehlerstandRechner

Field	Type	Null	Key	Default	Extra
<i>Id</i>	int(11)	NO	PRI	NULL	auto_increment
EingangsWertNr	int(11)	NO	MUL	NULL	
AusgangsWertNr	int(11)	NO	MUL	NULL	
ZeitstempelAnfang	datetime	NO		current_timestamp()	
Anfangsstand	decimal(10,3)	NO		NULL	
Intervall	int(11)	NO		NULL	
ZeitType	varchar(20)	NO	MUL	NULL	

In der Tabelle ZaehlerstandRechner werden alle Informationen gespeichert, die für die Berechnung des Verbrauchs im Intervall Stunden/Tage/Wochen/Monat/Jahr erforderlich sind. EingangsWertNr und AusgangsWertNr müssen gültige Werte aus der Wertetabelle sein. ZeitstempelAnfang gibt den Zeitstempel an, an dem der Anfangswert gespeichert wurde, für Monatswerte wäre dies der erste Tag des Monats. Intervall und Zeittyp gibt an, in welchem Intervall die Daten berechnet werden sollen.

ZeitTypen

Field	Type	Null	Key	Default	Extra
ZeitType	varchar(20)	NO	PRI	NULL	
Beschreibung	varchar(30)	NO		NULL	

Die einzige Aufgabe der Tabelle ZeitTypen besteht darin, die Tabelle ZaeherstandsRechner mit gültigen Zeittypen zu versorgen, d.h. mit in Maria-Db gültigen Werten wie „Day“, „Week“ oder „Year“.

Aktuell

Field	Type	Null	Key	Default	Extra
Zeit	datetime	NO	PRI	current_timestamp()	
WertNr	int(11)	NO	PRI	NULL	
Wert	decimal(12,2)	NO		NULL	

In der Aktuell-Tabelle wird für jeden Wert der aktuelle bzw. der letzte gültige Wert gespeichert. Dabei wird der Wert durch seine WertNr identifiziert. Jede WertNr darf hier nur einmal vorkommen. Für jeden Wert wird auch der Zeitstempel der Aktualisierung gespeichert.

Verlauf

Field	Type	Null	Key	Default	Extra
Zeit	datetime	NO	PRI	current_timestamp()	
WertNr	int(11)	NO	PRI	NULL	
Wert	decimal(12,2)	NO		NULL	

Die aktuelle Tabelle und die Verlaufstabelle unterscheiden sich im Aufbau nicht. In der Verlaufstabelle werden alle Werte gespeichert, die über einen längeren Zeitraum aufgezeichnet werden sollen. Jeder Eintrag wird durch eine Kombination aus Zeit und WertNr identifiziert, was den Vorteil hat, dass jeder Wert nur einmal mit genau dem gleichen Zeitstempel in der Tabelle vorkommen kann.

6.3 Views

Da die Tabellen Verlauf und Aktuell nur aus Zeit, WertNr und Wert bestehen, kann bei der Betrachtung schnell der Überblick verloren gehen. Um die Betrachtung und den Zugriff zu erleichtern, wurden SQL Views erstellt, mit ihnen werden die Tabellen über Fremdschlüssel zusammengeführt und als eine Tabelle angezeigt, nach außen sieht diese Tabelle wie eine echte aus, auf der Datenbank existiert sie aber nur virtuell. Die Views erleichtern auch die spätere PHP-Abfrage der Webseite, so können alle wichtigen Informationen mit einem Select-Befehl aus der View-Tabelle geholt werden und müssen nicht mühsam über viele Join-Befehle in der Anwendung zusammengesetzt werden. Aktuell und Verlauf View sind identisch aufgebaut.

Aufbau

Field	Type	Null	Key	Default	Extra
WertNr	int(11)	NO		0	
Zeit	datetime	YES		current_timestamp()	
Bereich	varchar(50)	NO		NULL	
Beschreibung	varchar(50)	NO		NULL	
Wert	decimal(12,2)	YES		NULL	
Einheit	varchar(10)	NO		NULL	
Medium	varchar(10)	NO		NULL	

Beispiel

WertNr	Zeit	Bereich	Beschreibung	Wert	Einheit	Medium
1	2023-06-22 09:59:47	Altbau	Wirkleistung_Gesamt	27.59	kW	Strom
2	2023-06-22 09:59:47	Altbau	Wirkleistung_L1	11.43	kW	Strom
3	2023-06-22 09:59:47	Altbau	Wirkleistung_L2	9.36	kW	Strom
4	2023-06-22 09:59:47	Altbau	Wirkleistung_L3	6.79	kW	Strom
5	2023-06-22 09:59:48	Cafetaria	Verbrauch_Monat	1513.31	kWh	Strom
6	2023-06-22 09:59:48	Cafetaria	Verbrauch_Jahr	7358.13	kWh	Strom
7	2023-06-22 09:59:49	Altbau Obergeschoss	Verbrauch_Monat	716.90	kWh	Strom
8	2023-06-22 09:59:49	Altbau Obergeschoss	Verbrauch_Jahr	5815.95	kWh	Strom
9	2023-06-22 09:59:49	Gesamt	Verbrauch_Monat	9593.00	kWh	Strom
10	2023-06-22 09:59:49	Gesamt	Verbrauch_Jahr	43026.00	kWh	Strom

WertNr	Zeit	Bereich	Beschreibung	Wert	Einheit	Medium
105	2023-06-22 09:59:50	Gesamt	Durchfluss	134.00	l/h	Wasser
21	2023-06-22 09:59:47	Lueftung	Wirkleistung_Gesamt	9.75	kW	Strom
11	2023-06-22 09:59:47	Cafetaria	Wirkleistung_Gesamt	4.48	kW	Strom
1	2023-06-22 09:59:47	Altbau	Wirkleistung_Gesamt	27.59	kW	Strom
105	2023-06-22 09:58:50	Gesamt	Durchfluss	55.00	l/h	Wasser
21	2023-06-22 09:58:47	Lueftung	Wirkleistung_Gesamt	9.77	kW	Strom
11	2023-06-22 09:58:47	Cafetaria	Wirkleistung_Gesamt	3.60	kW	Strom
1	2023-06-22 09:58:46	Altbau	Wirkleistung_Gesamt	28.82	kW	Strom
105	2023-06-22 09:57:49	Gesamt	Durchfluss	137.00	l/h	Wasser
21	2023-06-22 09:57:46	Lueftung	Wirkleistung_Gesamt	9.78	kW	Strom

7. Webseite

Um die gesammelten Daten übersichtlich darstellen zu können, wurde ein Dashboard erstellt. Als Webserver wird Apache verwendet. Über diesen ist die Seite erreichbar. Auf der Seite gibt es neben der Hauptseite (Dashboard) noch Unterseiten für die einzelnen Medien (Strom, Wasser, Gas). Auf den Unterseiten bekommt man genauere Informationen zu den einzelnen Medien und kann über eine Datumssuche die Verlaufsdaten eingrenzen, außerdem wurde mit Hilfe der Apache Authentifizierung ein gesicherter Bereich geschaffen, in dem Konfigurationen wie Zähler hinzufügen oder Abfragefelder anpassen möglich sind. Die Idee dahinter ist, dass alle notwendigen Konfigurationen zentral über die Webseite vorgenommen werden können. Zusätzlich soll ein Excel-Export möglich sein, um die gespeicherten Daten auch Externen zur Verfügung zu stellen. Außerdem wurde darauf geachtet, dass die Seite responsiv ist, so dass sie auf allen Endgeräten gut dargestellt wird.

7.1 Dashboard

Im oberen Bereich des Dashboards befindet sich die Navigationsleiste, mit der zu den Unterseiten und zum Admin Bereich gewechselt werden kann.

Darunter befindet sich der Hauptteil, dieser ist in Kacheln aufgeteilt, in jeder Kachel können beliebige Dinge dargestellt werden, z.B. über Grafiken oder Tabellen. So bleibt die Seite modular. Diese Kacheln werden mit der CSS-Grid-Methode realisiert. Die Seite verwendet JavaScript fetch, um Daten von verschiedenen APIs zu erhalten. So muss die Seite nicht jedes Mal neu geladen werden, um neue Daten zu erhalten.

In der oberen linken Kachel befindet sich ein Balkendiagramm, das den Verlauf des Stromverbrauchs anzeigt, wobei die angezeigten Werte alle 30 Sekunden nach dem Schema Stunde/Tag/Woche/Monat/Jahr gewechselt werden. Oben in der Mitte befindet sich eine Tabelle mit den Verbrauchswerten für alle Medien und Zeiträume. Je nachdem, in welchem Schema sich das rechte Diagramm befindet, ist die erste Spalte farbig umrandet, z.B. ist bei angezeigten Tageswerten das Feld Tag umrandet. Oben links wird das Wetter der EST Wetterstation angezeigt. Darunter befindet sich eine Kachel, die die aktuelle Netzauslastung im Bereich Tettnang/Bodenseekreis anzeigt. Diese Information wird über die StromGedacht API zur Verfügung gestellt. Ändert sich die Auslastung, wechselt die Ampel die Farbe und der Text darunter wird angepasst. Rechts daneben befindet sich ein Liniendiagramm, das den Strommix in Deutschland anzeigt (Strommix = Anteile der Energiequellen, z.B. Solar, Wind, Biogas); diese Informationen werden von der Bund-API SMARD zur Verfügung gestellt. Um diesen Graphen besser lesen zu können, hat er die breite von zwei Kacheln. Alle Grafiken werden mit der JavaScript-Bibliothek ChartJs erzeugt. Wie genau die Daten abgerufen und verwendet werden, wird später erklärt. Auf der nächsten Seite ist das Dashboard zu sehen.

Energiespiegel

[Dashboard](#) [Strom](#) [Wasser](#) [Gas](#) [Login](#)

Stromverbrauch

Zeit	Strom (kWh)	Gas (m³)	Wasser (m³)
13.06	~100	~0	~0
14.06	~100	~0	~0
15.06	~100	~0	~0
16.06	~100	~0	~0
17.06	~100	~0	~0
18.06	~100	~0	~0
19.06	~100	~0	~0
20.06	~100	~0	~0
21.06	~100	~0	~0

Gesamtwerte

Zeit	Strom	Solar	Gas	Wasser
Stunde	37 kWh	1.44 kWh	0 m³	0.2 m³
Tag	370 kWh	6.3 kWh	0 m³	1.2 m³
Woche	2472 kWh	74.03 kWh	0 m³	11.3 m³
Monat	9689 kWh	547.23 kWh	5 m³	108.8 m³
Jahr	43122 kWh	1625.28 kWh	3162 m³	185.4 m³

Wetter in Tuttlingen

Außentemperatur: 24.6 °C
Innentemperatur: 30.8 °C
Luftfeuchtigkeit Außen: 72%
Sonneninstrahlung: 659.0 W/m²
Uv-Index: 5.4
Regenrate: 0.0 mm/h
Regenmenge: 0.0
Windrichtung: WSW
Windgeschwindigkeit: 30.8 km/h
Letzte Aktualisierung: 2023-06-22 11:38:35

Netzauslastung

Normalbetrieb – Du musst nichts weiter tun

Strommix Deutschland

Zeit	Steinkohle (Gesamt Netzauf)	Pumpspeicher Verbrauch	Biomasse	Wasserkraft	etc. Konventionelle	Wind Offshore	Erdgas	Pumpspeicher	Wind Onshore	Photovoltaik	etc. Erneuerbare	Kernenergie
12.06.23	~70 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
13.06.23	~60 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
14.06.23	~50 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
15.06.23	~40 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
16.06.23	~30 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
17.06.23	~20 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0
18.06.23	~10 GWh	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0	~0

Energiespiegel

Dashboard Strom Wasser Gas Login

Stromverbrauch



Datum	Stromverbrauch (kWh)
29.05.23	~2000 kWh
05.06.23	~2000 kWh
12.06.23	~3800 kWh

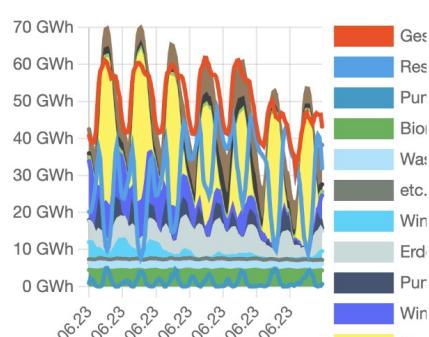
Gesamtwerte

	Strom	Solar	Gas	Wasser
Stunde	42 kWh	1.66 kWh	0 m ³	0.2 m ³
Tag	375 kWh	6.52 kWh	0 m ³	1.2 m ³
Woche	2477 kWh	74.25 kWh	0 m ³	11.3 m ³
Monat	9694 kWh	547.45 kWh	5 m ³	108.8 m ³
Jahr	43127 kWh	1625.5 kWh	3162 m ³	185.4 m ³

Wetter in Tettnang

Außentemperatur: 24.8 °C
 Innentemperatur: 30.8 °C
 Luftfeuchtigkeit Außen: 71%
 Sonneneinstrahlung: 763.0 W/m²
 Uv-Index: 5.8
 Regenrate: 0.0 mm/h
 Regenmenge: 0.0
 Windrichtng: W
 Windgeschwindigkeit: 30.8 km/h
 Letzte Aktualisierung: 2023-06-22 11:43:36

Strommix Deutschland



Zeitraum	Ges	Res	Pur	Bio	Wa	etc.	Win	Erd	Pur	Win	Phc
12.06.23 - 18.06.23	~60 GWh	~10 GWh	~5 GWh	~2 GWh	~1 GWh	~1 GWh	~10 GWh	~5 GWh	~1 GWh	~1 GWh	~10 GWh

Netzauslastung

Netzauslastung



Normalbetrieb – Du musst nichts weiter tun

7.2 Unterseiten

Die Unterseiten sind im Wesentlichen wie das Dashboard aufgebaut. Hervorzuheben ist die Filterkachel, die eine Suche nach einem bestimmten Zeitraum oder nach der Anzahl der Stunden/Tage/Wochen usw. ermöglicht.

Daten Filtern

Filtern nach: Auflösung:
Letzte:

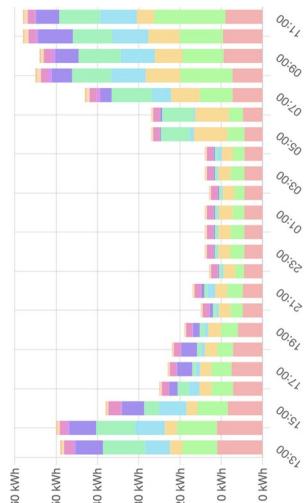
Daten Filtern

Filtern nach: Auflösung:
 bis

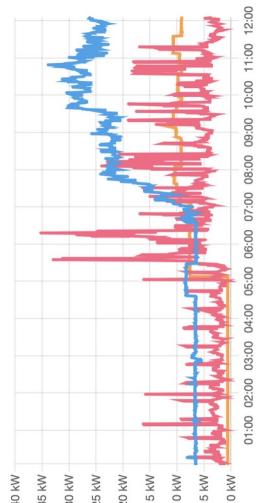
Energiespiegel

Dashboard Strom Wasser Gas Login

Strom Verbrauch



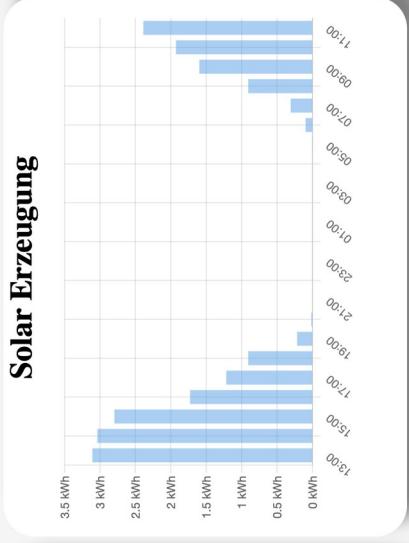
Leistung Vergangene 12 Stunden



Gesamt Verbrauch

Verbrauch	
Stunde	4 kWh
Tag	395 kWh
Woche	2497 kWh
Monat	9714 kWh
Jahr	43147 kWh

Daten Filtern



Aktuelle Stunde | Bereich

Bereich		Aktuelle Stunde
Altbau Obergeschoss		0.64 kWh
IT		0.38 kWh
Heizung		0.11 kWh
Altbau Sonstiges		0.61 kWh
Metallwerkstatt		0.33 kWh
Elektrowerkstatt UV1		0.04 kWh
Elektrowerkstatt UV2		0.01 kWh
Lüftung		0.62 kWh
Cafeteria		0.21 kWh
Neubau Sonstiges		3.1 kWh
Solar		0.16 kWh

7.3 Admin-Bereich

Der Administrationsbereich ist durch die Apache-eigene Authentifizierung geschützt. Um diese zu nutzen, muss sie in der Serverkonfiguration aktiviert werden und eine Passwortliste erstellt werden. Neue Benutzer können der Liste mit htpasswd hinzugefügt werden. Innerhalb des Root-Ordners der Webseite muss ein Unterverzeichnis angelegt werden. Innerhalb der Virtual Host Konfigurationsdatei muss dieses Verzeichnis zusammen mit der erzeugten Passwortliste angegeben werden. Nach einem Neustart des Apache Servers kann auf dieses Verzeichnis erst nach erfolgreicher Anmeldung mit den zuvor definierten Anmeldedaten über den Browser zugegriffen werden. Innerhalb dieses Ordners können nun Seiten erstellt werden, die vor unbefugtem Zugriff geschützt sind.

Um die Programmierung zu vereinfachen und einen zu großen Overhead zu vermeiden, wurde die Administrationsseite in PHP ohne Javascript Fetch-Aufrufe erstellt. Die Interaktivität erfolgt hier über HTML-Formulare. So kommuniziert die Seite mit Hilfe von PHP direkt mit der Datenbank ohne den Umweg über ein zusätzliches PHP API. Auf der Seite werden alle aktuellen Werte in einer Tabelle angezeigt, dazu werden die Werte mit Hilfe von PHP aus der Datenbank ausgelesen und in eine HTML-Tabelle eingefügt. Es ist auch möglich, neue Zähler hinzuzufügen. Hierfür wird ein HTML-Formular zur Verfügung gestellt, über Eingabefelder werden die neuen Informationen eingegeben und anschließend mit der Post-Methode an den Server gesendet. Auf dem Server wird es von PHP überprüft und in die Datenbank eingefügt.

7.4 ChartJs

Die Diagramme auf der Website wurden mit ChartJs erstellt. ChartJs ist eine Java Script Bibliothek zur Visualisierung von Daten. Dazu wird ein Chart Objekt erzeugt und diesem ein HTML Canvas Objekt zugeordnet. In diesem Canvas-Objekt wird dann das Chart gezeichnet. Canvas-Objekte dienen unter anderem dazu, Grafiken auf Webseiten mit Hilfe von JavaScript zu erzeugen.

Mit ChartJs können viele verschiedene Diagrammtypen erzeugt werden, neben Linien-, Kreis- und Balkendiagramme auch Polar-Area-Diagramme oder Radardiagramme.

Die Bedienung ist einfach gehalten. Zu Beginn wird dem erzeugte Char-Objekt das besagte Canvas-Objekte mitgegeben, zusätzlich werden ein Config-Block und ein Data-Block definiert und übergeben.

Jeder dieser Blöcke ist ein JavaScript-Objekt, dadurch ist die Anpassung sehr einfach, jeder Parameter ist hier ein Schlüssel-Werte-Paar.

Im Config-Block wird das Diagramm konfiguriert, hier wird z.B. angegeben, um welchen Typ es sich handeln soll, außerdem werden hier die Achsen definiert.

Im Data-Block werden Informationen zu den einzelnen Daten bereitgestellt. Dabei gibt es als Oberpunkt das Dataset, in dem sich alle Datenpunkte befinden. Zu jedem Datenpunkt können Informationen wie z.B. Bezeichnung, Hintergrundfarbe oder Sichtbarkeit angegeben werden. Außerdem werden hier auch die eigentlichen Daten in Form eines Arrays übergeben.

Da alle Parameter in einem JS-Objekt gespeichert werden, können jederzeit Parameter mit Hilfe des Punktoperators editiert oder hinzugefügt werden. Mit char.update wird dann das Diagramm innerhalb des Canvas-Objektes neu berechnet und auf der Seite angezeigt.

Auf diese Weise ist es sehr einfach, neue Daten, die gerade aus der Datenbank kommen, in ein bestehendes Chart einzufügen, ohne die Seite neu laden zu müssen.

Data-Block

```
//Data Block
const data = {
  labels: [],
  datasets: [
    {
      label: 'Solar',
      pointRadius: 1,
      data: [],
      borderColor: 'rgb(255, 243, 64)',
      backgroundColor: 'rgba(255, 243, 64, 1)'
    },
    {
      label: 'Gesamt',
      pointRadius: 1,
      data: [],
      hidden: false,
      borderColor: 'rgba(255, 0, 0, 0.5)',
      backgroundColor: 'rgba(255, 0, 0, 0.5)'
    }
  ]
};
```

7.5 Daten Abrufen

Um die Daten aus der Datenbank zu erhalten, werden die PHP-APIs mittels JavaScript aufgerufen. Als Antwort wird ein JSON mit den Daten zurückgegeben.

Aktuell

Um die Aktuellen Werte zu erhalten wird die Adresse „<http://ess.elektronikschule.de/api/aktuell.php>“ aufgerufen. Beim Aufruf wird serverseitig das Skript aktuell.php ausgeführt. Dieses holt die Werte aus der viewAktuell und speichert sie in einem zweidimensionalen Array. Für jede empfangene Wertereihe wird ein Datenpunkt an das Array angehängt. Anschließend wird das Array in einen Json-String konvertiert und zurückgegeben. Der empfangene String muss nur noch mittels JavaScript in ein Json- bzw. JS-Objekt dekodiert werden, danach werden die Werte den entsprechenden Charts/Tabellen zugewiesen. Die API wurde bewusst offen gestaltet, so dass mit jeder Programmiersprache oder einfach direkt über den Browser darauf zugegriffen werden kann.

Beispiel Rückgabe

```
[{"WertNr": "1", "Wert": "21.38", "Zeit": "2023-06-22  
14:51:47", "Bereich": "Altbau", "Beschreibung": "Wirkleistung_Gesamt", "Einheit": "kW", "Medium": "Strom"},  
 {"WertNr": "2", "Wert": "7.19", "Zeit": "2023-06-22  
14:51:47", "Bereich": "Altbau", "Beschreibung": "Wirkleistung_L1", "Einheit": "kW", "Medium": "Strom"},  
 {"WertNr": "3", "Wert": "6.57", "Zeit": "2023-06-22  
14:51:47", "Bereich": "Altbau", "Beschreibung": "Wirkleistung_L2", "Einheit": "kW", "Medium": "Strom"},  
 {"WertNr": "4", "Wert": "7.62", "Zeit": "2023-06-22  
14:51:47", "Bereich": "Altbau", "Beschreibung": "Wirkleistung_L3", "Einheit": "kW", "Medium": "Strom"},
```

Verlauf

Um verlaufswerte zu erhalten muss die Adresse [http://ess.elektronikschule.de/api/verlauf.php?WertNr=\[1\]&StartZeit=\[2\]&EndZeit=\[3\]](http://ess.elektronikschule.de/api/verlauf.php?WertNr=[1]&StartZeit=[2]&EndZeit=[3]) aufgerufen werden. Für das erste Feld muss die gewünschte WertNr eingegeben werden. Für das zweite und dritte Feld muss der Zeitraum in Form eines Unix-Timestamps übergeben werden, für den die Daten gewünscht werden. Dabei gibt das zweite Feld den Startzeitpunkt und das dritte den Endzeitpunkt an.

Nach dem Aufruf der Adresse wird das Skript verlauf.php ausgeführt. Es wird geprüft, ob und welche Parameter angegeben sind. Anstelle von WertNr kann auch nach dem Medium selektiert werden. Sind nicht alle Parameter angegeben, wird ein leerer Jason String zurückgegeben. Mit den übergebenen Parametern werden die Werte aus der Datenbank geholt. Dabei werden für jeden Wert die zugehörigen Werte und Zeiten in ein separates Array gepackt. Anschließend werden die Daten in einen Json String umgewandelt und zurückgegeben. Mittels Javascript muss die Antwort nur noch konvertiert werden, dank der Aufteilung in Arrays können die Werte direkt an das Chart Objekt übergeben werden. Auch diese API ist offen und kann mit entsprechenden Parametern aufgerufen werden.

Beispiel Rückgabe

```
[{"WertNr":"59","Wert":["0.97","2.38","2.39","2.39","1.93","1.60","0.91","0.31","0.10","0.01","0.00","0.00","0.00","0.00","0.00","0.00","0.00","0.00","0.02","0.22","0.91","1.22","1.73"],"Zeit":["2023-06-22 14:00:00","2023-06-22 13:00:00","2023-06-22 12:00:00","2023-06-22 11:00:00","2023-06-22 10:00:00","2023-06-22 09:00:00","2023-06-22 08:00:00","2023-06-22 07:00:00","2023-06-22 06:00:00","2023-06-22 05:00:00","2023-06-22 04:00:00","2023-06-22 03:00:00","2023-06-22 02:00:00","2023-06-22 01:00:00","2023-06-22 00:00:00","2023-06-21 23:00:00","2023-06-21 22:00:00","2023-06-21 21:00:00","2023-06-21 20:00:00","2023-06-21 19:00:00","2023-06-21 18:00:00","2023-06-21 17:00:00","2023-06-21 16:00:00"]}]
```

7.6 Externe Quellen

Um das Dashboard interessanter zu gestalten, werden zusätzliche externe Daten angezeigt. Diese Daten werden von den einzelnen Anbietern über eine API zur Verfügung gestellt. Eine API (Application Programming Interface) ist eine Schnittstelle, über die Anbieter Daten zur Verfügung stellen können. Der Aufruf erfolgt auch hier über eine URL, welche Parameter zu übergeben sind, kann der Dokumentation der jeweiligen API entnommen werden.

StromGedacht API

StromGedacht ist eine Anwendung der TransnetBW GmbH. TransnetBW ist der Übertragungsnetzbetreiber in und für Baden-Württemberg. Mit der App können sich Nutzer vor drohenden Netzengpässen warnen lassen. Ähnlich einer Ampel informiert die App über die aktuelle Netzauslastung. Insgesamt gibt es vier Stufen.

1. grüner Zustand: Normalbetrieb – Du musst nichts weiter tun²
2. gelber Zustand: Verbrauch vorverlegen – Strom jetzt nutzen²
3. oranger Zustand: Verbrauch reduzieren, um Kosten und CO2 zu sparen²
4. roter Zustand: Verbrauch reduzieren, um Strommangel zu verhindern²

Die Idee hinter dieser Anwendung ist es, gemeinsam möglichen Netzengpässen frühzeitig vorzubeugen.

TransnetBW stellt diese Daten nun über eine API zur Verfügung. Die Nutzung dieser API ist denkbar einfach. Sie wird unter der Adresse

,,https://api.stromgedacht.de/v1/now?zip=88069“ aufgerufen, wobei für zip die Postleitzahl des jeweiligen Ortes angegeben werden muss. Anschließend erhält man einen Json-String, in dem der jeweilige Status für den übermittelten Ort zurückgegeben wird: {"state":1}. Im Fall des Dashboards wird diese API alle 15 Minuten im Hintergrund abgefragt, und sobald sich der Status ändert, wird die Ampel auf der Webseite umgeschaltet und der Text darunter in den entsprechenden Text geändert. Um die Farbe der Ampel zu ändern, wird einfach die Farbe des entsprechenden Status dunkler und die anderen Farben heller gemacht, so dass ein guter Kontrast untereinander entsteht. Vielen Dank an TransnetBW für die Bereitstellung dieser genialen und einfachen API(<https://www.stromgedacht.de/>)

² Quelle: <https://www.stromgedacht.de/api-docs>

SMARD API

SMARD steht für Strommarktdaten. SMARD ist eine Informationsplattform der Bundesnetzagentur, auf der wichtige Informationen über das deutsche Stromnetz veröffentlicht werden, u.a. der deutschlandweite Stromverbrauch und die Stromerzeugung, aufgeteilt nach Erzeugungsquellen (Solar, Erdgas, Biomasse etc.). Unter „<https://www.smard.de>“ können diese Daten, neben anderen interessanten Netzdaten, visuell angesehen werden.

Die Bundesstelle für Open Data bietet eine API für diese Daten an. Beim Abruf der Daten sind einige Dinge zu beachten.

Zunächst wird die Adresse

„[https://www.smard.de/app/chart_data/\[Filter\]/DE/index_hour.json](https://www.smard.de/app/chart_data/[Filter]/DE/index_hour.json)“ aufgerufen, wobei eine gültige Schlüsselzahl für den [Filter] angegeben werden muss. Jeder Energieträger hat seine eigene Nummer. Braunkohle hat die Nummer 1223, Solar die Nummer 4068.

Weitere Informationen gibt es in der offiziellen Dokumentation

(<https://github.com/bundesAPI/smard-api>).

Als Rückgabewert dieses Aufrufs erhält man einen JSON-String mit Unix-Timestamps, die angeben, für welche Zeitpunkte Werte für die entsprechende Erzeugerquelle vorliegen.

Diese Zeitstempel wiederholen sich im Wochen Rhythmus.

Anschließend ruft man die Adresse

„[https://www.smard.de/app/chart_data/\[Filter\]/DE/\[Filter\]_DE_hour_\[Zeit\].json](https://www.smard.de/app/chart_data/[Filter]/DE/[Filter]_DE_hour_[Zeit].json)“ auf. Als [Filter] wird wieder die Schlüsselzahl gewählt. Als [Zeit] wird einer der zuvor erhaltenen Zeitstempel verwendet. Als Antwort erhält man einen Json-String mit einem zweidimensionalen Array, wobei jeder Index ein Array mit zwei Werten besitzt, bestehend aus dem Zeitstempel und dem zugehörigen Wert. Diese haben eine Auflösung von einer Stunde.

Um auf dem Dashboard ein ähnliches Diagramm wie auf der SMARD-Website anzuzeigen, werden alle Erzeugungsquellen abgerufen. Dazu müssen die obigen Schritte für jede Schlüsselzahl wiederholt werden. Um den Zugriff über JavaScript zu erleichtern, wurde ein PHP-Skript zwischengeschaltet. Strommarkt.php führt die obigen Schritte für alle gültigen Schlüsselzahlen aus und erhält so alle wichtigen Daten. Anschließend werden die Daten sortiert und in einem mehrdimensionales Array gespeichert, wobei jeder Energieträger seinen eigenen Bereich mit allen zugehörigen Werten hat. Dieses Array wird dann in einen Json-String gepackt und zurückgegeben/ausgegeben. Somit muss das Dashboard nur die Adresse „<http://ess.elektronikschule.de/api/strommarkt.php>“ aufrufen und erhält mit einem Aufruf alle wichtigen Informationen, die dann einfach an den entsprechenden Graphen übergeben werden können.

Vielen Dank auch an dieser Stelle an die Bundesstelle für Open Data für die Bereitstellung dieser API, diese und weitere APIs können unter „<https://bund.dev/apis>“ eingesehen werden.

8. Wetterstation

Bei der Suche nach einem Temperatursensor zur Anzeige der Außentemperatur auf dem Dashboard kamen wir auf die Idee, diese Daten von der Wetterstation zu übernehmen. Dies ist naheliegend, da die EST schon seit vielen Jahren im Besitz dieser Wetterstation ist. Bei der Begutachtung fiel auf, dass die Daten relativ einfach von der Wetterstation in eine Datenbank gesendet werden, so dass am Ende nicht nur die Temperatur auf dem Dashboard angezeigt wird, sondern auch alle anderen Daten der Wetterstation. Dazu wurde die Wetterstation so konfiguriert, dass sie ihre Daten an den bereits vorhandenen Datenbankserver sendet. Aus Sicherheitsgründen wurde für die Wetterstation eine zusätzliche Datenbank erstellt, die parallel zur Energiedatenbank läuft. Diese besitzt eine Tabelle, in der die Werte gespeichert werden.

Field	Type	Null	Key	Default	Extra
zeit	datetime	NO		current_timestamp()	
outdoortemp	decimal(4,1)	NO		NULL	
outdoorhumidity	int(5)	NO		NULL	
sealevelpressure	decimal(5,1)	NO		NULL	
solarrad	decimal(5,1)	NO		NULL	
uvindex	decimal(4,1)	NO		NULL	
rainrate	decimal(4,1)	NO		NULL	
rainfall	decimal(5,1)	NO		NULL	
winddirection	varchar(3)	NO		NULL	
windspeed	decimal(4,1)	NO		NULL	
windchill	decimal(4,1)	NO		NULL	
indoortemp	decimal(4,1)	NO		NULL	

Die Daten werden alle 5 Minuten von der Wetterstation an die Datenbank gesendet. Um die Daten für die Webseite aus der Datenbank zu erhalten, wurde wiederum eine Schnittstelle mit PHP geschrieben. Dabei holt `wetter.php` die letzten 300 Messwerte aus der Datenbank der Wetterstation. Anschließend werden die Daten in einen Json-String gepackt und zurückgegeben/ausgegeben.

Die Daten werden über „<http://ess.elektronikschule.de/api/wetter.php>“ aus dem Dashboard ausgelesen. Da es aufgrund von Verbindungsproblemen (Funkverbindung zwischen Wetterstation, Steuerung und den eigentlichen Sensoren) vorkommen kann, dass für eine gewisse Zeit falsche Daten von der Wetterstation in die Datenbank geschrieben werden, werden die Werte zunächst auf Plausibilität geprüft (bei falschen Werten wird -999 in die Tabelle geschrieben). Anschließend werden alle Datenpunkte durchlaufen und der letzte gültige Wert auf dem Dashboard angezeigt.

9. Sicherheit

Bei jedem Schritt wurde auf die Sicherheit geachtet. Angefangen beim Python-Programm. Da die Kommunikation über eine serielle Schnittstelle unter Linux besondere Rechte erfordert, wurde ein Benutzer angelegt, der nur die nötigsten Rechte besitzt. Für den Datenbankzugriff hat jeder Endpunkt seine eigenen Zugangsdaten. Jeder Endpunkt hat nur Zugriff auf die Tabellen, auf die er wirklich zugreifen muss, und darf nur die Kommandos (Select/Insert/Update) ausführen, die er für seine Arbeit benötigt.

10. Optimierungen

Obwohl das System wie vorgesehen funktioniert, gibt es noch einige Punkte, die optimiert werden können.

M-Bus Schnittstelle

Um die Ausfallsicherheit zu erhöhen könnte noch eine zentrale Fehlerprotokollierung implementiert werden. So das eventuelle Fehler z.B. bei der Zählerabfrage auch auf der Webseite angezeigt werden können.

Außerdem wäre es noch interessant weitere M-Bus Funktionen zu implementieren, so dass bestimmte Felder direkt abgefragt werden können. Auch eine automatische Dekodierung der Daten anhand der DIF und VIF Felder wäre denkbar.

Datenbank

Die Datenbank kann dahingehend optimiert werden, dass alle Anfragen über Prepared Statements laufen, was die Sicherheit enorm erhöhen und zusätzlich eine weitere Kontrollsicht bieten würde. Außerdem sollte über eine Backup-Strategie nachgedacht werden.

Website

Die Website kann noch weiter verbessert werden. Grundlegende Abläufe können noch effizienter gestaltet werden. Außerdem könnte das Design noch angepasst werden. Außerdem könnten noch mehr Datenpunkte visuell dargestellt werden. Außerdem ist mir aufgefallen, dass der Bund neben der SMART API noch weitere Schnittstellen mit interessanten Informationen zur Verfügung stellt. Unter anderem eine Schnittstelle zum Warnsystem NINA, über die Informationen zur aktuellen Gefahrenlage abgerufen werden können. Auch die Einbindung der Wetterstation ist ausbaufähig. Dies wurde jedoch vorerst bewusst vernachlässigt, da dies nicht der eigentliche Schwerpunkt dieser Arbeit war. Es wäre jedoch schade, diese Daten nicht zu nutzen.

11. Fazit

Anfangs hatte ich Bedenken, dass der Umfang dieser Arbeit zu gering sein könnte. Mit dem Fortschreiten der Arbeit wurden diese Zweifel jedoch nach und nach ausgeräumt. Spätestens nach der Entwicklung des Python-Programms, das viele Tassen Kaffee gekostet hat, waren alle Zweifel ausgeräumt. Auch die Datenbank erwies sich als komplexer als erwartet. Auch das Design und der Aufbau der Website waren relativ aufwendig. Allerdings lief nicht alles nach Plan. So wurde ein großer Teil der Zeit mit unnötigen Arbeiten verbracht. Zum Beispiel wurde, nachdem die ersten Daten ausgelesen werden konnten, schnell eine einfache Datenbank erstellt und die Daten darin gespeichert. Als dann das Ausleseprogramm fertig war, habe ich gemerkt, dass die Datenbank so auf Dauer nicht funktioniert. Also musste ein neues Design her. Nachdem dieses entwickelt war und zum ersten Mal zum Einsatz kam, mussten noch alle bis dahin angefallenen Daten in das neue Konzept eingepflegt werden. Als das erledigt war und ich mit dem Aufbau der Website begann, stellte ich fest, dass ich vergessen hatte, Optionen für die Steuerung der Zähler einzubauen. So musste ich die Zählerabfrage noch einmal überarbeiten und anschließend die gesamte Datenbank an die neuen Gegebenheiten anpassen. Dies führte wiederum dazu, dass alle Datenpunkte erneut angepasst werden mussten.

Mit etwas besserer Planung hätte man diese unnötigen Dinge vermeiden und viel Zeit sparen können.

Diese Zeit hat mir am Ende gefehlt, um die letzten Punkte perfekt zu machen. So funktioniert das ganze Projekt zwar gut und zuverlässig, aber es gibt noch einige Punkte auf meiner Liste, die ich gerne umsetzen würde.

Trotzdem bin ich mit dem Verlauf der Arbeit sehr zufrieden. Auch die Dinge, die ich bei dieser Arbeit gelernt habe, waren enorm. Außerdem war es eine gute Übung, die im Unterricht gelernten Dinge anzuwenden, egal ob Netzwerktechnik, Datenbanken, Computersysteme oder Softwareentwicklung, von jedem Thema war etwas dabei. Bei der Erstellung dieser Dokumentation musste ich immer wieder schmunzeln, als ich die ganzen Arbeitsschritte noch einmal durchgegangen bin, ich war überrascht, dass ich diese Sachen erstellt habe. Wenn mir vor Beginn der Technikerausbildung jemand gesagt hätte, dass ich in 1,5 Jahren so etwas auf die Beine stellen kann, hätte ich ihn für verrückt erklärt. Ich bin froh diesen Weg eingeschlagen zu haben.

Nach der Abgabe dieser Dokumentation wird dieses Projekt aber nicht einfach so zu Ende gehen, sondern ich möchte auf jeden Fall noch die offenen Punkte in die Tat umsetzen.

Abschließend möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit mit Rat und Tat unterstützt haben. Ein besonderer Dank gilt Herrn Rixner für seine stete Hilfsbereitschaft sowie für die Unterstützung beim Aufbau und der Verkabelung der einzelnen Messstellen.

Ich versichere, dass die Technikerarbeit von mir selbstständig angefertigt und nur die angegebenen Hilfsmittel benutzt wurden.

Alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, habe ich durch Angabe der Quellen kenntlich gemacht.

Datum

Unterschrift

